

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-142918

(43)Date of publication of application : 25.05.2001

(51)Int.Cl.

G06F 17/50

H01L 21/82

H01L 27/04

H01L 21/822

(21)Application number : 11-321346

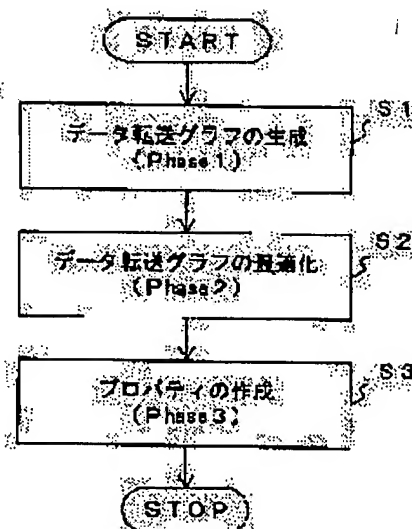
(71)Applicant : FUJITSU LTD

(22)Date of filing : 11.11.1999

(72)Inventor : SUBIRU ROI
IWASHITA HIROAKI
NAKADA TSUNEO

(54) DEVICE AND METHOD FOR GENERATING PROPERTY TO BE VERIFIED ABOUT HARDWARE

(57)Abstract:

PROBLEM TO BE SOLVED: To provide a method for automatically generating a property to be verified about hardware described in a hardware description language.**SOLUTION:** Data resources and a data transfer described in a hardware description language are expressed using a data transfer display graph (step S1). The data transfer graph is optimized (step S2). The data transfer graph is simplified or minimized by optimizing processing. On the basis of the topology of the optimized data transfer graph, the property to be verified by a verifying tool is generated (step S3). The property to be verified is resource contention and register omission.プロパティ生成ソールの
動作を説明するフローチャート

LEGAL STATUS

[Date of request for examination] 13.11.2003

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the
examiner's decision of rejection or application converted
registration]

[Date of final disposal for application]

[Patent number] 3943299

[Date of registration] 13.04.2007

[Number of appeal against examiner's decision of
rejection][Date of requesting appeal against examiner's decision of
rejection]

[Date of extinction of right]

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2001-142918

(P2001-142918A)

(43) 公開日 平成13年5月25日 (2001.5.25)

(51) Int.Cl. ⁷	識別記号	F I	テームト [*] (参考)
G 0 6 F 17/50		G 0 6 F 15/60	6 7 0 Z 5 B 0 4 6
H 0 1 L 21/82			6 6 4 A 5 F 0 3 8
27/04			6 7 0 K 5 F 0 6 4
21/822		H 0 1 L 21/82	T
			C

審査請求 未請求 請求項の数17 O L (全 41 頁) 最終頁に続く

(21) 出願番号	特願平11-321348	(71) 出願人	000005223 富士通株式会社 神奈川県川崎市中原区上小田中4丁目1番1号
(22) 出願日	平成11年11月11日 (1999.11.11)	(72) 発明者	スビル ロイ 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内
		(72) 発明者	岩下 洋哲 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内
		(74) 代理人	100074099 弁理士 大曾 義之 (外1名)

最終頁に続く

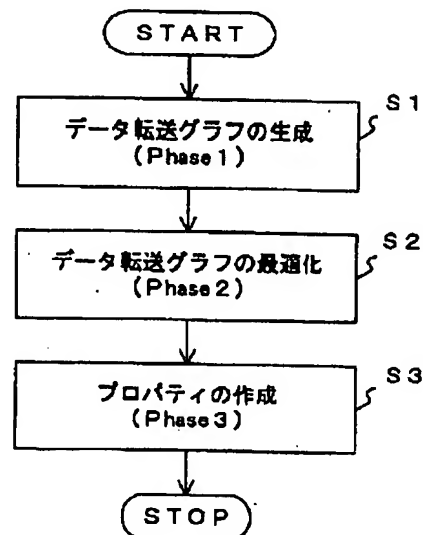
(54) 【発明の名称】 ハードウェアについて検証すべきプロパティを生成する装置および方法

(57) 【要約】

【課題】 ハードウェア記述言語により記述されているハードウェアについて検証すべきプロパティを自動的に生成する方法を提供する。

【解決手段】 ハードウェア記述言語により記述されているデータ資源およびデータ転送をデータ転送表示グラフを用いて表す (ステップS1)。データ転送グラフを最適化する (ステップS2)。最適化処理により、データ転送グラフは簡略化または最小化される。最適化されたデータ転送グラフのトポロジに基づいて検証ツールにより検証すべきプロパティを生成する (ステップS3)。検証すべきプロパティは、資源競合およびレジスタ漏れである。

プロパティ生成ツールの
動作を説明するフローチャート



【特許請求の範囲】

【請求項1】 ハードウェア記述言語により記述された複数の資源およびそれらの資源を利用したデータ転送を含むハードウェアについて検証すべきプロパティを生成する装置であって、

上記ハードウェア記述言語で記述された資源およびデータ転送に対応するデータ転送グラフを生成するグラフ生成手段と、

そのグラフ生成手段により生成されたデータ転送グラフを最適化する最適化手段と、

その最適化手段により最適化されたデータ転送グラフを利用して、上記ハードウェア記述言語により記述されているハードウェアについて検証すべきプロパティを生成するプロパティ生成手段と、

を有するプロパティ生成装置。

【請求項2】 請求項1に記載のプロパティ生成装置であって、

上記最適化手段は、データ転送グラフのトポロジに基づいてそのグラフを最小化する。

【請求項3】 請求項1に記載のプロパティ生成装置であって、

上記グラフ生成手段は、各資源にそれぞれ対応するノード、および各データ転送の転送元および転送先を表すエッジを用いて上記ハードウェアを表すデータ転送グラフを生成する。

【請求項4】 請求項3に記載のプロパティ生成装置であって、

上記最適化手段は、互いに隣接する2つのノードに対応する各資源のビット幅が互いに同じであり、且つ一方の資源が他方の資源の子供である場合、それらのノードのうちの一方を上記データ転送グラフから削除する。

【請求項5】 請求項3に記載のプロパティ生成装置であって、

上記最適化手段は、第1の資源から第2の資源および第3資源にデータが転送され、それら第2および第3の資源のビット幅が上記第1の資源のビット幅よりも小さい場合、上記第1の資源に対応するノードを上記データ転送グラフから削除する。

【請求項6】 請求項3に記載のプロパティ生成装置であって、

上記最適化手段は、第1の資源および第2の資源から第3資源にデータが転送され、且つ上記第1および第2の資源のビット幅が上記第3の資源のビット幅よりも小さい場合、上記第3の資源に対応するノードを上記データ転送グラフから削除する。

【請求項7】 請求項3に記載のプロパティ生成装置であって、

上記最適化手段は、レジスタからそのレジスタの名称と同じ名称が付与されている出力ポートにデータが転送される場合、上記出力ポートに対応するノードを上記デ

ータ転送グラフから削除する。

【請求項8】 請求項3に記載のプロパティ生成装置であって、

上記最適化手段は、第1の資源から送出されるデータが第2の資源のみに転送され、且つ上記第2の資源が上記第1の資源から転送されてくるデータのみを受け取る場合、上記第1の資源から上記第2の資源へのデータ転送を表すエッジを上記データ転送グラフから削除する。

【請求項9】 請求項3に記載のプロパティ生成装置であって、

上記最適化手段は、第1の資源から第2の資源へのデータ転送と第1の資源から第3資源へのデータ転送との間で転送すべきビットがオーバーラップしない場合に、上記2つのデータ転送に対応するエッジを上記データ転送グラフから削除する。

【請求項10】 請求項3に記載のプロパティ生成装置であって、

上記最適化手段は、第1の資源から第2の資源へのデータ転送と第3の資源から第2資源へのデータ転送との間で転送すべきビットがオーバーラップしない場合に、上記2つのデータ転送に対応するエッジを上記データ転送グラフから削除する。

【請求項11】 請求項3に記載のプロパティ生成装置であって、

上記最適化手段は、複数のレジスタが同一のレジスタファイルに属する場合、それら複数のレジスタに対応する複数のノードを1つのノードに変換する。

【請求項12】 請求項1に記載のプロパティ生成装置であって、

上記プロパティ生成手段は、資源競合およびレジスタ漏れのうちの少なくとも1つに係わるプロパティを生成する。

【請求項13】 請求項1に記載のプロパティ生成装置であって、

上記プロパティ生成手段は、あるノードに複数のファンインエッジが接続されている場合、それら複数のファンインエッジに対応するデータ転送に対してそれぞれ与えられている条件に基づいて資源競合プロパティを生成する。

【請求項14】 請求項1に記載のプロパティ生成装置であって、

上記プロパティ生成手段は、あるノードにファンインエッジおよびファンアウトエッジが接続されている場合、それらのエッジに対応するデータ転送に対してそれぞれ与えられている条件に基づいてレジスタ漏れプロパティを生成する。

【請求項15】 ハードウェア記述言語により記述された複数の資源およびそれらの資源を利用したデータ転送を含むハードウェアについて検証すべきプロパティを生成する装置であって、

上記ハードウェア記述言語で記述された資源およびデータ転送に対応するデータ転送グラフを生成するグラフ生成手段と、

上記データ転送グラフのトポロジを利用して、上記ハードウェア記述言語により記述されているハードウェアについて検証すべきプロパティを生成するプロパティ生成手段と、

を有するプロパティ生成装置。

【請求項16】 ハードウェア記述言語により記述された複数の資源およびそれらの資源を利用したデータ転送を含むハードウェアについて検証すべきプロパティを生成する方法であって、

上記ハードウェア記述言語で記述された資源およびデータ転送に対応するデータ転送グラフを生成し、

そのデータ転送グラフを最適化し、

その最適化されたデータ転送グラフを利用して、上記ハードウェア記述言語により記述されているハードウェアについて検証すべきプロパティを生成するプロパティ生成方法。

【請求項17】 ハードウェア記述言語により記述された複数の資源およびそれらの資源を利用したデータ転送を含むハードウェアについて検証すべきプロパティを生成するためのプログラムを格納した記録媒体であって、上記プログラムがコンピュータにより実行されたときに、

上記ハードウェア記述言語で記述された資源およびデータ転送に対応するデータ転送グラフを生成する手段と、そのデータ転送グラフを最適化する手段と、

その最適化されたデータ転送グラフを利用して、上記ハードウェア記述言語により記述されているハードウェアについて検証すべきプロパティを生成する手段とを提供する記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、ハードウェア記述言語により記述された複数の資源およびそれらの資源を利用したデータ転送を含むハードウェアについて検証すべきプロパティを生成する装置および方法に係わる。

【0002】

【従来の技術】近年、半導体チップの設計は非常に複雑になってきている。すなわち、半導体チップ上に非常に複雑なハードウェア（論理装置）が実装されるようになってきている。このため、半導体チップの設計は、複数の設計者による共同作業で行われることが多い。

【0003】ところが、複数の設計者により細分化された設計情報は、各設計者により誤って解釈されることがある。この場合、ある一部における誤り（設計ミス）は、レジスタ転送レベルにおいて、システム全体が正しく動作しなくなる原因となることがある。そして、もし、そのような誤りがあると、デバッグ作業などに多大

な費用と時間が浪費されてしまう。

【0004】ところで、半導体チップ上に設けることができる資源（例えば、レジスタ、バス、入力ポート、出力ポートなど）は有限である。このため、各資源は、通常、複数のタスクにより共有される。そして、共有されている資源により生成されるデータは、しばしば、一時的にデータ記憶用資源（例えば、レジスタなど）に格納される。

【0005】このような状況において頻繁に発生する設計誤りは、「資源競合」および「レジスタ漏れ」である。資源競合は、1つの資源（レジスタなど）が複数の資源により同時にアクセスされる誤りである。一方、レジスタ漏れは、複数の資源によりあるレジスタがアクセスされる際のアクセス順番に係わる誤りである。

【0006】複雑なハードウェアが実装された半導体チップ上には、膨大な数の資源が形成されている。このため、上述のような誤り（資源競合、レジスタ漏れなど）を人手で全て発見することは実質的に不可能である。ところが、設計されたハードウェアの中にたった1つの誤りがあっただけでシステム全体が正しく動作しなくなってしまうこともある。したがって、すべての資源競合およびレジスタ漏れを発見して取り除いておくことは、非常に重要である。

【0007】

【発明が解決しようとする課題】多くの設計者は、設計作業の早い段階で資源競合やレジスタ漏れを発見することの重要性を認識している。しかし、現在までのところ、資源競合やレジスタ漏れを発見するための作業は、各設計者によりその設計者が担当するサブシステムごとに行われているに過ぎなかった。

【0008】システム全体の動作を保証するためには、サブシステムごとの検証だけでは十分ではなく、設計されたハードウェア全体に渡って資源競合やレジスタ漏れを探し、それらを取り除く必要がある。しかし、従来は、そのような作業はなされていなかった。このため、ハードウェア設計に誤りが含まれていた場合には、それらの誤りは、通常、半導体チップ上にそのハードウェア設計を実装した後に検出されることになる。

【0009】本特許出願の出願人が知る範囲では、資源競合およびレジスタ漏れを自動的に検出し、定式化し、確認するツールは存在しない。そして、そのようなツール無しでは、誤りのないハードウェア設計を実現することは実質的に不可能であろうと思われる。

【0010】本発明の課題は、ハードウェア記述言語により記述されているハードウェアについて検証すべきプロパティを生成することができる装置および方法を提供することである。

【0011】

【課題を解決するための手段】本発明のプロパティ生成装置は、ハードウェア記述言語により記述された複数の

資源およびそれらの資源を利用したデータ転送を含むハードウェアについて検証すべきプロパティを生成する構成を前提とし、以下の各手段を有する。グラフ生成手段は、ハードウェア記述言語で記述された資源およびデータ転送に対応するデータ転送グラフを生成する。最適化手段は、そのグラフ生成手段により生成されたデータ転送グラフを最適化する。プロパティ生成手段は、その最適化手段により最適化されたデータ転送グラフを利用して、上記ハードウェア記述言語により記述されているハードウェアについて検証すべきプロパティを生成する。

【0012】上記構成において、ハードウェア記述言語により記述されている設計情報がグラフ化され、そのグラフを利用してハードウェアについて検証すべきプロパティが自動的に生成される。このとき、プロパティが生成される前にデータ転送グラフが最適化されるので、プロパティの生成が容易になる。生成されたプロパティは、検証ツールにより検証される。

【0013】資源は、例えば、レジスタ、バス、入力ポート、出力ポートであり、データ転送グラフにおいてそれぞれノードで表される。一方、データ転送は、その転送元および転送先が定義されたエッジ（矢印）を用いて表される。そして、最適化手段は、データ転送グラフのトポロジに基づいてグラフを最小化する。

【0014】生成すべきプロパティは、例えば、資源競合およびレジスタ漏れである。資源競合のプロパティは、データ転送グラフにおいて、複数のファンインエッジが接続されたノードにおいて検出される。一方、レジスタ漏れは、データ転送グラフにおいて、ファンインエッジおよびファンアウトエッジが接続されたレジスタを表すノードにおいて検出される。

【0015】

【発明の実施の形態】以下、本発明の実施形態について図面を参照しながら説明する。図1は、本発明の一実施形態のプロパティ生成ツールが使用される環境を説明する図である。本実施形態のプロパティ生成ツール1は、予め作成されているハードウェア仕様2からプロパティを抽出し、それを検証ツール3に検証させるものである。

【0016】ハードウェア仕様2は、あるハードウェア（例えば、半導体チップ上に実装されるプロセッサ等の論理装置）の設計記述であり、そのハードウェアに含まれる複数の資源、およびそれら資源間のデータ転送を定義する。このハードウェア仕様2は、たとえば、レジスタ転送レベルのハードウェア記述言語で記述されている。ハードウェア記述言語は、例えば、Verilogである。また、定義される資源は、例えば、レジスタ、レジスタファイル、バス、入力ポート、出力ポート、入出力ポートなどである。

【0017】ハードウェア仕様2において記述されているハードウェアは、1以上のモジュールを含む。モジュ

ールが複数あるときは、それらは、しばしば階層的に記述される。この場合、しばしば、上位階層のモジュールを「親（ペアレント）」、下位階層のモジュールを「子供（チャイルド）」と呼ぶことがある。なお、各モジュールは、それぞれ1以上の資源を含む。

【0018】プロパティ生成ツール1は、入力されたハードウェア仕様2を解析し、プロパティ（検証ツール3が検証すべき項目または条件式）を抽出する。この実施例では、プロパティは、「資源競合」および「レジスタ漏れ」であり、例えば、CLTなどの時相論理式（時間経過に伴う信号変化の条件式）で表される。

【0019】検証ツール3は、既存の論理検証ツールであり、プロパティ生成ツール1により抽出されたプロパティについて検証する。具体的には、ハードウェア仕様2により記述されているハードウェアにおいて「資源競合」または「レジスタ漏れ」が発生するか否かを検証する。なお、検証ツール3は、他の項目についても検証することができる。

【0020】図2は、プロパティ生成ツール1の動作を説明するフローチャートである。この処理は、ハードウェア仕様2が入力されたときに実行される。ステップS1（Phase1）では、ハードウェア仕様2の記述に基づいて、データ転送グラフ（DTDAG：Data Transfer Directed Acyclic Graph）が作成される。このとき、ハードウェア仕様2において定義されている各モジュール毎に資源が認識され、さらに、モジュールが階層的に記述されているときはそれらの資源の親-子関係も認識される。また、データ転送に係わる条件が考慮される。これらの条件は、ハードウェア仕様2に記述されている。データ転送グラフは、ハードウェア仕様2により記述されているハードウェアに含まれる資源およびデータ転送を表す。

【0021】ステップS2（Phase2）では、ステップS1で作成されたデータ転送グラフが最適化（簡略化または最小化）される。具体的には、データ転送グラフにおいて表されている資源およびデータ転送のうち、「資源競合」又は「レジスタ漏れ」が発生する可能性のない資源またはデータ転送が削除される。

【0022】ステップS3（Phase3）では、ステップS2において最適化されたデータ転送グラフからプロパティを抽出し、さらにそれらに対応する入力スクリプトを生成する。この入力スクリプトは、検証ツール3が処理可能な形式で記述される。

【0023】次に、図3を参照しながら、データ転送グラフおよび抽出すべきプロパティについて説明する。図3は、データ転送グラフの一例である。このデータ転送グラフは、入力されたハードウェア仕様に基づいて作成されるが、ここでは、その作成方法の説明は省略する。なお、データ転送グラフは、基本的に、コンピュータ画面に実際に表示されるものではない。したがって、図3

に示すグラフは、データ転送グラフを模式的に示したものである。このことは、以下の図面においても同様である。ただし、データ転送グラフをコンピュータ画面に実際に表示することは可能であり、そのようにしてもよい。

【0024】各ノードA～F、H、Iは、それぞれデータ資源に対応する。例えば、ノードAは、ハードウェア仕様2において「interface control register file」という名称が与えられたレジスタである。また、ノード間に描かれている矢印は、データ転送に対応する。以下、この矢印を「データ転送エッジ」または単に「エッジ」と呼ぶことにする。また、例えば、ノードAからノードDへのデータ転送に対応するエッジは、ノードAにとってはファンアウトエッジ（または、出力エッジ）であり、同時に、ノードDにとってはファンインエッジ（または、入力エッジ）である。

【0025】各データ転送エッジには、そのデータ転送についての定義や条件が付与されている。「C」は、データ転送が「同時転送（Concurrent Transfer）」であることを表し、「S」は、そのデータ転送が「順次転送（Sequential Transfer）」であることを表す。また、「Gi」は、データ転送についての動作条件（guardor enabling condition）を表す。たとえば、図3に示す例では、条件G1が満たされたときに、ノードAからノードDへのデータ転送が実行される。

【0026】上記データ転送グラフにおいて、あるノードに対して複数のファンインエッジが存在する場合（Multiple fan-in）、「資源競合」が発生する可能性がある。ノードDに注目する。ノードDは32ビットのレジスタ「reg data out」である。ノードDには、エッジADおよびエッジBDが入力されている。したがって、条件G1が満たされたときに、ノードA（レジスタ「interface control register file」）に格納されているデータがノードD（レジスタ「reg data out」）に転送され、また、条件G2が満たされたときに、ノードB（入力ポート「data out」）からノードD（レジスタ「reg data out」）へデータが転送される。この場合、もし、任意のクロックサイクルにおいて、条件G1およびG2が同時に満たされると、レジスタ「reg data out」に対して同時に2つの書込アクセスが発生することになる。すなわち、「資源競合」が発生することになる。

【0027】上述のような「資源競合」は、データ転送グラフでは、複数のファンインエッジが存在するノードにおいて発生し得る。したがって、プロパティ生成ツール1は、データ転送グラフにおいて複数のファンインエッジが存在するノードを検出する。そして、検証ツール3が、そのノードに係わるデータ転送のタイミングを考慮しながら、実際に「資源競合」が発生するかどうかを検証する。

【0028】また、あるノードに対してファンインエッ

ジおよびファンアウトエッジが存在する場合には、「レジスタ漏れ」が発生する可能性がある。例えば、ノードDには、ファンインエッジAD、BD、およびファンアウトエッジDFが存在する。ここで、このハードウェアは、ノードAのデータをノードDを介してノードFに送った後に、ノードBのデータをノードDに書き込む動作をするものとする。したがって、この動作は、条件G1、G6、G2がその順番で満たされたときに得られる。ところが、もし、条件G6が満たされる前に条件G1および条件G2が満たされてしまうと、ノードDからノードFへのデータ転送は、ノードDがノードBから転送されてくるデータにより上書き（data overwrite）された後に実行されることになる。すなわち、「レジスタ漏れ」が発生することになる。

【0029】なお、「レジスタ漏れ」は、あるノードに対するファンインエッジが1つの場合にも起こり得る。例えば、第1のノードから第2のノードを介して第3のノードへデータを転送する場合に、第2のノードから第3のノードへのデータ転送が実行される前に、第1のノードから第2のノードへ他のデータが転送されると、上述の場合と同様に、「レジスタ漏れ」が起こり得る。

【0030】このように、上述のような「レジスタ漏れ」は、データ転送グラフにおいてあるノードに1以上のファンインエッジおよび1以上のファンアウトエッジが存在する場合に発生し得る。したがって、プロパティ生成ツール1は、データ転送グラフからファンインエッジおよびファンアウトエッジが存在するノードを検出する。そして、検証ツール3が、そのノードに係わるデータ転送のタイミングを考慮しながら、実際に「レジスタ漏れ」が発生するかどうかを検証する。

【0031】レジスタに対応するノードに対してファンインエッジおよびファンアウトエッジが存在する場合には、さらに他の問題が内在する。例えば、もし、あるレジスタに対応するノードのファンインエッジに係わる条件と、そのノードのファンアウトエッジに係わる条件とが同一クロックサイクル内で満たされるとすると、そのレジスタに対して書込アクセスおよび読出アクセスが同時に行われることになる。このような状況は、レジスタ転送レベルのモデルでは必ずしもエラーではない。しかし、この状況は、ゲートレベル或いはレイアウトレベルにおいてクロックの歪みを考慮すると、上記レジスタから読出したデータを読み出してしまふ恐れがある。すなわち、もし、クロックが歪んでいると、あるクロックサイクルにおいて、レジスタからのデータ読出しが完全に終わる前にそのレジスタに新たなデータの書込みが行われてしまうことがある。この問題は、実際の半導体チップ上では、レジスタ漏れプロパティとして現れる。なお、図3に示す例では、例えば、ノードIからノードCへのデータ転送とノードCからノードDへのデータ転送との間でこの種の「資源競合」が起こり得る。

【0032】上述のように、「資源競合」および「レジスタ漏れ」が実際に発生するの否かは、検証ツール3により検証される。すなわち、プロパティ生成ツール1は、これらの問題が発生する可能性のあるプロパティをハードウェア仕様2から自動的に生成し、それらを検証ツールに渡す。

【0033】なお、上述のプロパティ（「資源競合」および「レジスタ漏れ」）は、記述されたハードウェアの資源およびデータ転送がデータ転送グラフで表された場合、そのグラフのトポロジに関係する。即ち、データ転送グラフのトポロジを解析すれば、「資源競合」または「レジスタ漏れ」が発生する可能性を判断できる。このため、プロパティ生成ツール1は、データ転送グラフのトポロジから自動的にプロパティを生成できる。ただし、この場合、資源およびデータ転送に係わる情報は、レジスタ転送レベルコードから抽出されている必要がある。

【0034】ところで、ハードウェア仕様2において記述されているハードウェアが大規模になると、必然的に、その記述から生成されるデータ転送グラフは複雑になる。このため、プロパティ生成ツール1は、プロパティを正確に且つ短時間に生成するために、生成したデータ転送グラフを最適化（最小化または簡略化）する。最適化の方法としては、データ転送グラフのトポロジに基づく最適化（トポロジ最適化）、およびデータ転送の動作条件に基づく最適化（論理最適化）が考えられる。

【0035】図4は、データ転送グラフを最適化する処理のフローチャートである。この処理は、図2のステップS2に相当する。ステップS11では、「非フラグメント等価（Unfragmented Equivalence）」を検出する。ステップS12では、「フラグメント等価（Fragmented Equivalence）」を検出する。ステップS13では、「名称等価（Named Equivalence）」を検出する。ステップS14では「フラグメントエッジ（Fragmented Edges）」を検出する。ステップS15では、「等価エッジ（Edges in Equivalence）」を検出する。ステップS16では、「レジスタファイル（Register File）」を検出する。上記ステップS11～S16により、資源どうしの等価関係が検出される。そして、それらの等価関係を利用し、データ転送グラフから「資源競合」または「レジスタ漏れ」が発生する可能性のない部分を削除する。これにより、データ転送グラフが最適化（すなわち、最小化）される。

【0036】ここで、ステップS11～S16の処理のうちの幾つかについて簡単に説明をしておく。なお、ステップS11～S16の各処理は、後述フローチャートを参照しながら詳しく説明する。

【0037】図5は、ハードウェアの例である。図6は、図5に示すハードウェアを階層的に記述する方法を説明する図である。ハードウェア資源およびデータ転送

は、しばしば、図6に示すように、階層的に設計されて記述される。ここでは、「モジュール0」が上位階層であり、「モジュール1」および「モジュール2」が「モジュール0」の下位階層である。

【0038】このように、階層ごとにデータ資源およびデータ転送が記述される場合、同一の資源またはデータに対して異なる名称が付与されることが多々ある。例えば、図6に示す例においては、モジュール0への入力である“input 1”は、モジュール1においては“input 11”として記述されている。

【0039】ところで、ハードウェア記述言語では、モジュール間のデータ転送は、通常、入力ポートまたは出力ポートを用いて記述される。たとえば、モジュール0における“input 1”とモジュール1における“input 11”との対応関係は、しばしば、モジュール0の“input port 1”からモジュール1の“input port 11”へのデータ転送として表される。本実施形態では、このようなポート同士の対応関係を「ポート連携（Port Association）」と呼ぶ。そして、この「ポート連携」は、ハードウェア仕様2において定義されている。

【0040】また、ハードウェア記述言語では、あるモジュール内のレジスタに格納されているデータを他のモジュールへ転送する場合、そのデータは、出力ポートを介して転送されるように記述される。この場合、その出力ポートには、通常、上記レジスタに付与されている名称と同じ名称が付与される。本実施形態では、この対応関係を「名称連携（Named Association）」と呼ぶ。

【0041】図7は、「資源の等価」を利用してデータ転送グラフを最適化する例を示す図である。図7(a)は、ハードウェア仕様2において記述されている資源およびデータ転送の定義から生成されたデータ転送グラフを示す。この実施例では、モジュール1には、3つの資源（入力ポートSD-DQ-IN、レジスタdata-in、出力ポートdata-in）が設けられている。また、モジュール1の出力は、モジュール2において定義されているバスワイヤdata-buf-outに転送され、さらに、モジュール3において定義されているレジスタreg-data-outに転送される。

【0042】ここで、レジスタdata-in および出力ポートdata-in は、互いに「名称連携」の関係を有する。すなわち、これら2つの資源は、等価（名称等価）であるものとみなすことができる。したがって、入力ポートSD-DQ-INからレジスタdata-inへのデータ転送は、入力ポートSD-DQ-INから出力ポートdata-inへのデータ転送として扱うことができる。また、出力ポートdata-in およびバスワイヤdata-buf-outは、互いに「ポート連携」の関係を有する。即ち、これら2つの資源は、等価であるものとみなすことができる。従って、入力ポートSD-DQ-INから出力ポートdata-inへのデータ転送は、入力ポートSD-DQ-INからバスワイヤdata-buf-outへのデータ転送

として扱うことができる。そして、これらの等価関係を利用すれば、図7(a)に示すデータ転送グラフは、図7(b)に示す形態に最適化(すなわち、簡略化)される。

【0043】なお、データ転送の転送元の資源と転送先の資源とが等価であるためには、それらの資源が扱うデータのビット幅が互いに同じである必要がある。従って、以下の定義が得られる。定義1：互いに隣接する1組の資源が互いに同じビット幅仕様を備える場合、それらの等価な資源は、「非フラグメント等価」である。

【0044】図8は、「フラグメント」を説明する図である。図8に示す例では、中間レベルのモジュールCに32ビット幅の出力ポートOPが設けられ、その上位レベルのモジュールBにそれぞれ8ビット幅の出力ポートOP1~OP4が設けられている。

【0045】ここで、あるデータ幅を持ったデータ資源とそのデータ幅よりも小さいデータ幅のデータ資源(あるいは、そのデータ幅よりも大きいデータ幅のデータ資源)との間でデータ転送がある場合、「フラグメント等価」が生じ得る。図8に示す例では、モジュールBの出力ポートOP1~OP4は、モジュールCの出力ポートOPとの関係において「フラグメント等価」とみなされる。

【0046】図8に示すモデルにおける出力ポートOP、および出力ポートOP1~OP4は、データ転送グラフでは、それぞれノードで表され、また、出力ポートOPから出力ポートOP1~OP4へのデータ転送は、それぞれエッジで表される。ここで、上述したように、出力ポートOP1~OP4にそれぞれ対応するノードが「フラグメント等価」である場合、上記データ転送に対応する4本のエッジはフラグメントエッジである。本実施形態では、「フラグメント等価」であるノードに接続される複数のエッジを、「フラグメントエッジ」と呼ぶ。

【0047】「フラグメント等価」の有用性は、フラグメントデータ転送を組み合わせることにより非フラグメントデータ転送を得る可能性があることである。また、データ転送の転送元資源または転送先資源が「フラグメント等価」であるか否かに基づいて、データ転送がフラグメントデータ転送であるか非フラグメントデータ転送であるかを識別することができる。

【0048】フラグメントデータ転送を最適化する他の方法は、「独立フラグメントデータ転送」である。この「独立フラグメントデータ転送」について、以下の定義を設ける。定義2：フラグメントデータ転送エッジの中の任意のエッジの組合せにおいて資源競合を起こさないのであれば、それらのエッジは「独立」である。

【0049】本実施形態の最適化処理では、データ転送グラフから「独立」なエッジが削除される。これにより、資源競合プロパティまたはレジスタ漏れプロパティが誤って生成されることが回避される。

【0050】レジスタファイルは、ハードウェア規模を小さくするために、複数のレジスタをグループ化し、バスと各レジスタとの間の接続を簡略化する技術である。これにより、例えば、図9(a)に示す構成が、図9(b)に示す構成に簡略化される。ただし、レジスタファイルに属する各レジスタは、ハードウェア記述言語では、しばしば独立したレジスタとして記述される。この場合、レジスタファイルは、データ転送グラフにおいて、複数のレジスタとして表わされる。このため、データ転送グラフの最適化処理では、レジスタファイルを認識し、そのレジスタファイルに属する複数のレジスタに対応する複数のノードを、1つのノードに置き換える。

【0051】次に、具体的な実施例を示す。

第1の実施例

図10~図15は、プロパティ生成ツール1に入力されるハードウェア仕様2の一例である。ここでは、ハードウェア記述言語として、Verilogが使用されている。

【0052】図10は、モジュール0について記述したコードである。このコードのセクション1の記述によれば、モジュール0は、2つの入力ポート(INPUT1, 2)、および2つの出力ポート(OUTPUT1, 2)を備える。これらの各ポートは、それぞれ16ビット幅である。また、モジュール0には、クロックおよび制御信号が入力される。さらに、モジュール0には16ビット幅のバスワイヤCONEXTが設けられている。

【0053】セクション2の記述によれば、モジュール0の下位階層にモジュール1が設けられている。そして、モジュール0とモジュール1の間におけるポートの対応関係(ポート連携)が定義されている。この例では、例えば、モジュール0の入力ポートINPUT1の8~15ビットがモジュール1の入力ポートIPOINT3に対応付けられ、モジュール0の入力ポートINPUT1の0~7ビットがモジュール1の入力ポートIPOINT4に対応付けられている。この場合、入力ポートINPUT1は、入力ポートIPOINT3および入力ポートIPOINT4の「親資源」であり、また、入力ポートIPOINT3および入力ポートIPOINT4は、入力ポートINPUT1の「子供資源」である。また、バスワイヤCONEXTがモジュール1の入力ポートIPOINT1, 2に接続されている。

【0054】セクション3の記述によれば、モジュール0の下位階層にモジュール2が設けられている。セクション3では、モジュール0とモジュール2の間におけるポートの対応関係が定義されている。また、バスワイヤCONEXTがモジュール2の出力ポートOPOINT1に接続されている。

【0055】図11および図12は、モジュール1について記述したコードである。セクション4の記述によれば、各入力ポート(IPOINT1-4)、出力ポート(OPOINT1, 2)はそれぞれ8ビット幅である。また、セクション5の記述によれば、モジュール1には、レジスタAおよび

レジスタBが設けられている。そして、レジスタAに格納されているデータは出力ポートOPORT1に転送され、レジスタBに格納されているデータは出力ポートOPORT2に転送される。なお、これらのデータ転送は、クロックまたは制御信号に係わらず実行される。すなわち、これらのデータ転送は、同時転送（Concurrent Transfer）タイプに属する。具体的には、“continuousassignment”による転送である。

【0056】セクション6の記述によれば、制御信号1が与えられると、入力ポートIPORT1に入力されたデータがレジスタAに書き込まれると共に、入力ポートIPORT3に入力されたデータがレジスタBに書き込まれる。また、セクション7の記述によれば、制御信号2が与えられると、入力ポートIPORT2に入力されたデータがレジスタAに書き込まれると共に、入力ポートIPORT4に入力されたデータがレジスタBに書き込まれる。これらのデータ転送は、制御信号に従って実行される。すなわち、これらのデータ転送は、順次転送（Sequential Transfer）タイプに属す。具体的には、“sequential always block”による転送である。

【0057】図13～図15は、モジュール2について記述したコードである。セクション8の記述によれば、各入力ポート（IPORT5）、出力ポート（OPORT3,4）はそれぞれ16ビット幅である。また、セクション9の記述によれば、モジュール2は、レジスタC～Fを備える。そして、レジスタEに格納されているデータは出力ポートOPORT3に転送され、レジスタFに格納されているデータは出力ポートOPORT4に転送される。これらのデータ転送は、同時転送である。また、各レジスタC～Fには、それぞれ所定の初期値が与えられている。

【0058】セクション10の記述によれば、制御信号3が与えられると、入力ポートIPORT5に入力されたデータがレジスタCに書き込まれ、そうでない場合には、入力ポートIPORT5に入力されたデータはレジスタDに書き込まれる。また、セクション11の記述によれば、制御信号4が与えられると、レジスタC、Dに所定の演算結果データが書き込まれる。さらに、セクション12の記述によれば、レジスタFにも所定の演算結果データが書き込まれる。

【0059】セクション13の記述によれば、制御信号5が与えられると、レジスタCに格納されているデータがレジスタEに書き込まれる。また、制御信号5が与えられることなく制御信号6が与えられると、レジスタDに格納されているデータがレジスタEに書き込まれる。さらに、制御信号5および6が与えられることなく制御信号7が与えられ場合、および制御信号5～7が与えられなかった場合は、それぞれレジスタEに所定の定数データが書き込まれる。

【0060】図16は、図10～図15に示したハードウェア仕様から生成されたデータ転送グラフである。こ

のグラフは、図2に示したフローチャートのステップSにより生成される。なお、ハードウェア仕様において定義されているデータ資源は、データ転送グラフでは、「ノード」として表される。そして、各ノードごとに、データ資源の名称、データ資源の種別（レジスタ、バスワイヤ、ポートなど）、モジュール名、およびノード名が設定される。また、各データ転送は、データ転送グラフでは、それぞれエッジ（図16では、「矢印」で描かれている）で表される。

【0061】なお、このデータ転送グラフは、発明を理解しやすくするために模式的に描いたものであり、実際は、モジュールリスト、データ資源リスト、データ転送エッジリスト、および等価クラスリスト等から構成される。

【0062】モジュールリストは、ハードウェア仕様において定義されている各モジュールについて以下の情報を格納する。

- ・モジュールの名称
- ・親モジュール（当該モジュールの上位階層のモジュール）
- ・子供モジュール（当該モジュールの下位階層のモジュール）
- ・階層レベル（当該モジュールが属する階層）
- ・処理状態（最適化処理における中間状態を表すフラグ等）
- ・各種ポインタ

例えば、モジュール0についてモジュールリストに格納すべき情報としては、ハードウェア仕様から以下が得られる。

- 【0063】
- ・モジュールの名称：MOD0
 - ・親モジュール：なし
 - ・子供モジュール：MOD1, MOD2
 - ・階層レベル：0

なお、「処理状態」は、最適化処理における中間状態を表すフラグ等であり、ハードウェア仕様から得られるのではない。また、「ポインタ」も、ハードウェア仕様から得られるのではない。

【0064】データ資源リストは、ハードウェア仕様において定義されている各データ資源について以下の情報を格納する。このリストは、資源の種別（レジスタ、入力ポート、出力ポート、バス、データ転送、定数、...）ごとに生成される。

- 【0065】
- ・資源の名称
 - ・親モジュール
 - ・階層レベル
 - ・ビット幅仕様
 - ・親資源
 - ・子供資源

- ・非フラグメント等価クラス
- ・フラグメント等価クラス
- ・エッジ
- ・処理状態

例えば、入力ポートINPUT1（ノードA）についてデータ資源リストに格納すべき情報としては、ハードウェア仕様から以下が得られる。

【0066】

- ・資源の名称：INPUT1
- ・親モジュール：なし
- ・階層レベル：0
- ・ビット幅仕様：「15：0」
- ・親資源：なし
- ・子供資源：IPORT5
- ・エッジ：エッジAF

なお、「非フラグメント等価クラス」および「フラグメント等価クラス」は、ハードウェア仕様から得られるのではなく、データ転送グラフを解析することにより得られるものである。

【0067】データ転送エッジリストは、ハードウェア仕様において定義されている各データ転送について以下の情報を格納する。

- ・エッジの名称
- ・転送先資源
- ・転送元資源
- ・ガード表現（データ転送の条件）
- ・データ転送タイプ（同時転送または順次転送など）
- ・転送先または転送元のフラグメント状態
- ・最適化処理における処理状態

例えば、入力ポートIPORT1（ノードJ）からレジスタREG-A（ノードO）へのデータ転送に対応するエッジについてデータ転送エッジリストに格納すべき情報としては、ハードウェア仕様から以下が得られる。

【0068】

- ・エッジの名称：JO
- ・転送先資源：レジスタREG-A（ノードO）
- ・転送元資源：入力ポートIPORT1（ノードJ）
- ・ガード表現：CTRL1
- ・データ転送タイプ：S（順次転送）

なお、「転送先または転送元のフラグメント状態」は、データ転送グラフを解析することにより得られる。

【0069】等価クラスリストは、以下の情報を格納する。なお、これらの情報は、データ転送グラフを解析することにより得られる。

- ・等価クラス識別子
- ・各等価クラスに属する資源
- ・処理状態

データ転送グラフを作成する際には、まず、ハードウェア仕様の各モジュール毎の記述を上位階層から順番に解析してゆき、定義されている各データ資源、それら資源

の対応関係、およびそれらの資源を利用したデータ転送を認識する。このとき、あるモジュールから他のモジュールへのコールがあるか否かを調べるために、behavioral 記述がスキャンされる。そして、そのようなコールがあった場合には、呼び出されたモジュール毎に、サーチキューを設け、先に認識されている資源との間でインタフェース定義の対応関係を認識する。

【0070】また、各ポート（入力ポート、出力ポート、入出力ポートを含む）、バスワイヤ、レジスタのビット幅を認識し、さらに、各バスワイヤごとに、そのバスワイヤに対応するドライバおよびレジスタを認識する。

【0071】ハードウェア資源2から抽出した情報は、モジュールリスト、データ資源リスト、データ転送エッジリストに格納される。図10～図15に示す例では、これらの処理により、図16に示す各ノードA～Y（E、G、P、Q、Rを除く）が生成され、それぞれビット幅が設定される。

【0072】さらに、データ転送毎に設定されている条件を解析する。これらの条件は、たとえば、“gurded continuous assignment statement”、“combinational always block”、“sequential always block”である。これらの情報は、「ガード表現」として、データ転送毎にデータ転送エッジリストに登録される。これにより、各データ転送に対応するエッジが生成されることになる。なお、条件が設定されていないデータ転送については、そのデータ転送に対して「1」が付与される。

【0073】ハードウェア仕様において代入式が記述されている場合（例えば、セクション10およびセクション11）には、図17(a)に示すように、疑似資源Sdtが生成される。この疑似資源Sdtは、記述されている代入式を実装する仮想的なデータソースである。たとえば、セクション11に記述されている代入式に対応するノードP（Sdt3）が生成される。ノードE、Gも同様である。また、あるレジスタに予め決められた定数が書き込まれる場合（例えば、セクション13）は、図17(b)に示すように、疑似資源Sdcが生成される。この疑似資源Sdcもまた仮想的なデータソースである。例えば、セクション13に記述された定義に対応してノードQ、Rが生成される。なお、これらの疑似資源は、ハードウェア仕様に基づいてデータ資源リストに所定の情報を登録することにより生成される。

【0074】続いて、上述のようにして生成されたデータ転送グラフが最適化される。図16に示す例では、データ転送グラフにおいて「非フラグメント等価」「フラグメント等価」および「独立エッジ」を検出することによりデータ転送グラフが最適化される。

【0075】「非フラグメント等価」を検出する場合、ハードウェア仕様のインタフェース定義において互いに「等価」とであると定義されている1組の資源であって、

データ幅が互いに同じであるものを認識する。この場合、対象となる資源は、たとえば、入力ポート、出力ポートおよび入出力ポートである。そして、非フラグメント等価クラスに属する資源が検出されると、そのことを表す情報が、データ資源リストにおいて、その資源に対応する記憶領域に登録される。

【0076】上記等価関係は、各ポート毎に調べられる。このとき、各ポート毎に処理は、最終資源にたどり着くまで、あるいは異なるビット幅の資源に到達するまで継続される。

【0077】ここで、図16に示したグラフにおける「非フラグメント等価」を説明する。ノードA（モジュール0のINPUT2）及びノードF（モジュール2のIPOINT5）は、共に16ビットの入力ポートである。また、これらの入力ポートは、ハードウェア仕様のセクション3に定義されているように、互に対応する資源である。即ち、ノードFに対応する資源は、ノードAに対応する資源の「子供」である。従って、ノードAおよびノードFは、「非フラグメント等価」である。同様に、ノードCはノードDと等価であり、ノードXはノードWと等価である。

【0078】この場合、例えば、資源Aおよび資源Fが非フラグメント等価クラス#1に属しているとする、データ資源リストにおいて、資源Aおよび資源Fの「非フラグメント等価クラス」にそれぞれ「#1」が登録される。

【0079】「フラグメント等価」を検出する場合は、分岐されている資源の中から等価なものを認識する。図16に示す例では、例えば、ノードBからノードHおよびノードIへデータが転送されている。このとき、ノードBのビット仕様が16ビット幅であるのに対し、ノードHおよびノードIのビット仕様はそれぞれ8ビット幅である。即ち、各転送先のノードで使用されるビット幅は、転送元のノードで使用されるビット幅よりも小さい。従って、ノードHおよびノードIは、「フラグメント等価」である。このことは、ノードJ、Kにおいても同様である。

【0080】一方、ノードYには、ノードSおよびノードTからデータが転送されてきている。このとき、各転送元のノードで使用されるビット幅は、転送先のノードで使用されるビット幅よりも小さい。従って、ノードSおよびノードTは、「フラグメント等価」である。

【0081】この場合、例えば、資源Hおよび資源Iがフラグメント等価クラス#5に属しているとする、データ資源リストにおいて、資源Aおよび資源Fの「フラグメント等価クラス」にそれぞれ「#5」が登録される。

【0082】「独立エッジ」は、以下のようなものである。たとえば、ノードPはノードUのみにデータを転送し、ノードUはノードPのみからデータを受け取る。こ

の場合、ノードPからノードUへのデータ転送に対応するエッジは、独立エッジである。また、ノードYには、ノードS、Tからデータが転送されてきている。このとき、ノードSからは0~7ビットが転送され、ノードTには8~15ビットが転送されてきている。すなわち、これら2つのデータ転送においてデータビットはオーバーラップしていない。この場合、ノードSからノードYへのデータ転送に対応するエッジ及びノードTからノードYへのデータ転送に対応するエッジは、共に独立エッジである。図16に示すグラフでは、この他にも、例えば、ノードNからノードSへのデータ転送、ノードOからノードTへのデータ転送、ノードUからノードDへのデータ転送、ノードVからノードWへのデータ転送に対応するエッジもそれぞれ独立エッジである。

【0083】「独立エッジ」と関連して、「フラグメントエッジ」および「等価エッジ」の概念を導入する。

「フラグメントエッジ」とは、転送元の資源または転送先の資源がフラグメント等価クラスに属するエッジのことを指す。例えば、図16に示す例では、ノードJおよびノードKがフラグメント等価クラスに属しているので、この場合、エッジCJ、エッジCK、エッジJO、エッジKOは、この等価クラスに係わるフラグメントエッジである。なお、各エッジは、(1) 転送元および転送先の資源が共に「フラグメント等価」でない状態、(2) 転送元の資源のみが「フラグメント等価」である状態、(3) 転送先の資源のみが「フラグメント等価」である状態、(4) 転送元および転送先の資源が共に「フラグメント等価」である状態に分類される。なお、この分類は、エッジ毎に2ビットのフラグで表される。

【0084】「等価エッジ」とは、転送元の資源または転送先の資源がフラグメント等価クラスに属するノードであるエッジの集合である。例えば、ノードJ、Kはフラグメント等価クラスに属するので、この場合、エッジCJ、エッジCK、エッジJO、エッジKOが、この等価クラスに係わる等価エッジである。なお、独立エッジは、等価エッジに属するエッジの集合から除去される。

【0085】上述のようにして等価関係を検出した後、データ転送グラフから「資源競合」または「レジスタ漏れ」に関わりのない部分を除去する。具体的には、ノードAおよびノードFは「非フラグメント等価」である。したがって、ノードAが除去される。また、エッジVWが「独立エッジ」であり、ノードWおよびノードXは「非フラグメント等価」である。したがって、ノードW、Xが除去される。さらに、エッジPUおよびエッジUDがそれぞれ「独立エッジ」であり、ノードDおよびノードCは「非フラグメント等価」である。したがって、ノードP、U、Dが除去される。

【0086】また、ノードH、Iは、ノードBから見て「フラグメント等価」である。従って、ノードBが除去される。同様に、ノードCも除去される。また、ノード

10

20

30

40

50

S、Tは、ノードYから見て「フラグメント等価」である。従って、ノードYが除去される。さらに、エッジN SおよびエッジO Tは共に独立エッジである。したがって、ノードS、Tが除去される。

【0087】上述の処理の結果、図16に示すデータ転送グラフは、図18に示す状態に最適化される。上述の処理は、具体的には、データ資源リストおよびデータ転送エッジリスト等からエッジリストE_{final}を生成する処理に相当する。エッジリストE_{final}は、最適化処理の結果として得られるエッジのリストである。

【0088】最適化処理が終わると、図2のフローチャートのステップS3の処理が実行される。この処理は、図19に示すように、2つのステップからなる。ステップS21では、最適化されたデータ転送グラフから隣接ノードリストを作成する。続いて、ステップS22において、プロパティおよびそれに対応する検証スクリプトを生成する。本実施形態では、「資源競合」または「レジスタ漏れ」が発生し得るすべての条件を抽出する。具体的には、隣接ノードリストを利用して各ノード毎の入力エッジ数および出力エッジ数をカウントし、それに基づいてプロパティを生成する。そして、抽出した「資源競合」または「レジスタ漏れ」が発生し得る条件が、検証ツール3が実行可能なスクリプトに変換される。

【0089】図18に示すグラフからプロパティを生成する場合の処理を説明する。まず、最適化されたグラフから隣接ノードリストが作成される。隣接ノードリストは、データ転送グラフ上の各エッジ毎に、その転送元ノードおよび転送先ノードを検出することにより得られる。図20に隣接ノードリストの例を示す。なお、各エッジ毎の転送先ノードおよび転送元ノードに係わる情報は、最適化処理により得られるエッジリストE_{final}に格納されている。

【0090】続いて、隣接ノードリストを利用し、「資源競合」または「レジスタ漏れ」が発生する可能性があるノードを検出する。「資源競合」を検出する場合には、複数の入力エッジが存在するノードを検出する。図20に示す例では、ノードL、M、N、Oにおいてそれぞれ2本に入力エッジが存在し、ノードVにおいて4本のエッジが存在する。従って、これら5つのノードは、それぞれ「資源競合」が発生する可能性があるものとみなされる。

【0091】一方、「レジスタ漏れ」を検出する場合には、1以上の入力エッジが存在し、且つ1以上の出力エッジが存在するノードを検出する。図20に示す例では、ノードL、Mにおいて、2本の入力エッジおよび1本の出力エッジが存在する。したがって、これらのノードは、「レジスタ漏れ」が発生する可能性があるものとみなされる。なお、クロックの歪みを考慮すると、「レジスタ漏れ」の可能性のあるノード（レジスタ）は、「資源競合」が発生する可能性があるものとみなされ

る。

【0092】図21は、「資源競合」のプロパティの例である。これらのプロパティは、図18に示すデータ転送グラフから生成されたものである。「資源競合」のプロパティは、上述した5つのノードについてそれぞれ生成される。このとき、各ノードにおいて存在する入力エッジに付与されている条件が抽出され、それらの条件を組み合わせることによりプロパティが生成される。

【0093】例えば、ノードLについてのプロパティを生成する場合、エッジE_LおよびエッジF_Lに付与されている条件がグラフから抽出される。すなわち、エッジE_LおよびエッジF_Lからそれぞれ「制御信号CTRL4」および「制御信号CTRL3」が抽出され、それらを組み合わせることにより下記のプロパティが得られる。

【0094】AG (CTRL4, CTRL3)

検証ツール3は、このプロパティが入力されると、制御信号CTRL4 および制御信号CTRL3 が同時に発生するクロックサイクルが存在するか否かを検証する。そして、それらが同時に発生するクロックサイクルが存在する場合は、検証ツール3は、ノードLにおいて「資源競合」が発生するものと判断する。即ち、レジスタCに対して同時書込アクセスが発生するものとみなす。

【0095】なお、あるノードにおいて3本以上の入力エッジが存在する場合には、それらの中から任意の2本の入力エッジを抽出し、その2本の入力エッジに付与されている条件がグラフから抽出される。この処理は、すべての組合せについて行われる。例えば、ノードVは4本の入力エッジが存在するので、この場合、6通りの組合せ（エッジQVとエッジLV、QVとMV、QVとRV、LVとMV、LVとRV、MVとRV）が得られる。そして、各組合せごとにプロパティが生成される。

【0096】図22は、「レジスタ漏れ」のプロパティの例である。これらのプロパティもまた、図18に示すデータ転送グラフから生成されたものである。なお、この実施例で検出される「レジスタ漏れ」は、「書込み、読出し、書込み」であるはずのシーケンスが「書込み、書込み、読出し」というシーケンスで実行されてしまう場合を想定する。

【0097】ノードLに着目した場合、「レジスタ漏れ」が発生する可能性がある条件は、以下の4つである。すなわち、(1) ノードEからノードLへデータが転送された後に、ノードLからノードVへの転送が実行されることなく、再びノードEからノードLへデータが転送される場合、(2) ノードFからノードLへデータが転送された後に、ノードLからノードVへの転送が実行されることなく、再びノードFからノードLへデータが転送される場合、(3) ノードFからノードLへデータが転送された後に、ノードLからノードVへの転送が実行されることなく、ノードEからノードLへデータが転送される場合、(4) ノードEからノードLへデータが転送さ

れた後に、ノードLからノードVへの転送が実行されることなく、ノードFからノードLへデータが転送される場合である。なお、ノードMにおいても、基本的に同様にプロパティが生成される。例えば、(1)の場合のプロパティは、

EF (CTRL4/\EX (E (~CTRL5 U CTRL4)))

と表される。

【0098】検証ツール3は、このプロパティが入力されると、「制御信号CTRL4が発生するクロックサイクルの後に制御信号CTRL5が発生することなく制御信号CTRL4が発生するクロックサイクル」を探す。そして、そのような状況が存在するのであれば、検証ツール3は、ノードLにおいて「レジスタ漏れ」が発生するものと判断する。

【0099】図23は、クロックの歪みを考慮した場合におけるレジスタにおける「資源競合」のプロパティの例である。これらのプロパティもまた、図18に示すデータ転送グラフから生成されたものである。なお、この場合の「資源競合」は、あるクロックサイクルにおいて、書込アクセスおよび読出アクセスが発生する場合を想定する。

【0100】ノードLに着目した場合、「資源競合」が発生する可能性がある条件は、以下の2つである。すなわち、(1)ノードEからノードLへデータ転送と、ノードLからノードVへのデータ転送が動イルクロックサイクル内に実行される場合、および(2)ノードFからノードLへデータ転送と、ノードLからノードVへのデータ転送が動イルクロックサイクル内に実行される場合である。なお、ノードMにおいても基本的に同様にプロパティが生成される。

【0101】例えば、(1)の場合のプロパティは、AG (CTRL4, CTRL5)

として表される。検証ツール3は、このプロパティが入力されると、制御信号CTRL4および制御信号CTRL5が同時に発生するクロックサイクルを探す。そして、そのようなクロックサイクルが存在するのであれば、検証ツール3は、ノードLにおいて「資源競合」が発生するものと判断する。

第2の実施例

図24～図26は、プロパティ生成ツール1に入力されるハードウェア仕様2の一例である。この実施例でも、ハードウェア記述言語として、Verilogが使用されている。

【0102】セクション1においては、入力される制御信号、入力データおよび出力データが定義されている。また、セクション1の記述によれば、このモジュールには5つのレジスタ (BASES, ENDS, MODE, REG-ADR, REGA-OUT) が設けられている。セクション2には、制御信号CNT-REG-ACC および制御信号WRITE が与えられたときの

データ転送が記述されている。これらのデータ転送は、5ビットの制御信号REG-ADRの値により制御される。また、セクション3には、各種制御信号が与えられたときのレジスタREG-ADR または出力REGA-OUTへのデータ転送が記述されている。

【0103】図27は、図24～図26に示したハードウェア仕様から生成されたデータ転送グラフである。ハードウェア仕様からデータ転送グラフを生成する方法は、第1の実施例の場合と同じなので、ここではその説明を省略する。

【0104】図27に示すグラフにおいて、「G1」～「G11」は、データ転送のトリガとしての条件である。これらの条件を図28に示す。例えば、ノードAからノードDへのデータ転送は、セクション2の記述によれば、制御信号CNT-REG-ACC および制御信号WRITE が与えられ、且つレジスタREG-ADRの第4ビットが「1」であり、且つレジスタREG-ADRの第3～0ビットが「0001」であったクロックサイクルにおいて実行される。図28に示す「G1」は、この条件を記述している。

【0105】なお、各条件に含まれる要素の集合を「サポートセット」と呼ぶ。たとえば、条件G1のサポートセットは、7つの要素 (CNT-REG-ACC, WRITE, REG-ADR(4), REG-ADR(3), REG-ADR(2), REG-ADR(1), REG-ADR(0)) を含んでいる。

【0106】このデータ転送グラフを最適化する際には、「レジスタファイル」および「名称等価」の概念が利用される。「レジスタファイル」を検出する場合、まず、転送先がレジスタであるエッジまたは転送元がレジスタであるエッジを抽出する。そして、それらのエッジに付与されているサポートセットを比較し、それらが互いに同じであるエッジ同士をグループ化する。この場合、グループ化された複数のエッジにそれぞれ接続されるレジスタを「レジスタファイル」とみなす。

【0107】図29を参照しながら具体例を示す。ここでは、転送先ノードがレジスタであるエッジADに注目する。エッジADには、条件G1が付与されている。この場合、レジスタファイルを検出する場合には、まず、レジスタを表すノードへのファンインエッジの中から、G1のサポートセットと同じサポートセットを有するエッジを抽出する。条件G1のサポートセットは、図28に示すように、条件G2～G6とそれぞれ同じである。従って、レジスタを表すノードへのファンインエッジエッジのうち、条件G2～G6が設定されているエッジが抽出される。これにより、条件G1～G6が設定されているエッジがグループ化される。すなわち、エッジAD、AE、AFがグループ化される。したがって、これらのエッジの転送先である3つのノード (D、E、F) は、レジスタファイルに属するものとみなされる。

【0108】なお、上記の例では、レジスタへのファンインエッジに基づいてレジスタファイルを検出したが、

図29に示すように、レジスタからのファンアウトエッジを利用して同様の関係が得られる。

【0109】「名称等価」を検出する場合は、同一モジュール内のノードの中で同一の名称が付与されているノード（特に、レジスタおよび出力ポート）を抽出し、それらを等価な資源とみなす。図27に示す例では、レジスタに対応するノードJおよび出力ポートに対応するノードIの名称が共に「REGA-OUT」なので、これらのノードが等価であるとみなされる。

【0110】上述のようにして等価関係を検出した後、データ転送グラフから「資源競合」または「レジスタ漏れ」に関わりがない部分を除去する。すなわち、ノードD、E、Fは、「レジスタファイル」なので、これらのノードは1つのノード（ノードD'）に置き換えられる。また、ノードJおよびノードIは「名称等価」である。したがって、ノードIが除去される。上述の最適化処理の結果、図27に示すデータ転送グラフは、図30に示す状態に最適化される。

【0111】データ転送グラフが最適化されると、それに伴って各エッジに付与されている条件（サポートセット）も変化する。具体的には、図27に示されていた条件G1～G6が条件G1'に集約され、一方、条件G7およびG8が条件G2'に集約される。図31に条件G1'および条件G2'のサポートセットを示す。

【0112】第2の実施例のハードウェアでは、ノードJにおいて「資源競合」が発生する可能性がある。この場合、「資源競合」のプロパティは、4つの条件（G2'、G9、G10、G8）の中の任意の2つが同時に発生するクロックサイクルを調べるためのスクリプトである。また、このハードウェアでは、ノードD'において「レジスタ漏れ」が発生する可能性がある。この場合、「レジスタ漏れ」のプロパティは、条件G1'および条件G2'が所定の順番で発生することを調べるためのスクリプトである。

【0113】次に、上述したプロパティ生成ツール1の動作アルゴリズムをフローチャートを参照しながら説明する。なお、以下の説明では、下記の定義を用いる。

- L モジュールの階層レベル
- M モジュールのセット
- Si 入力ポートのセット
- So 出力ポートのセット
- Sio 入出力ポートのセット
- Sw バスワイヤのセット
- Sr レジスタのセット
- Sdt データ要素（代入式等の演算の結果を生成する資源）
- Sdc データ定数（定数を生成する資源）
- Edt データ転送エッジ
- N ノードのセット
- H データ資源の等価クラスのセット

図32～図34は、データ転送グラフを生成する方法を説明するフローチャートである。この処理は、ハードウェア記述言語で記述されたハードウェア仕様2が入力されたときに実行される。

【0114】ステップS31では、プロパティを作成するために利用するメモリ領域を初期化する。このとき、モジュール階層Lとして「0」が設定される。また、最小ビット幅パラメータとして「Wb」が設定される。

【0115】ステップS32では、モジュールリストMにルートモジュールを登録する。モジュールリストMには、ハードウェア仕様2に設けられているすべてのモジュールが登録されている。ステップS33は、モジュールリストMに登録されている各モジュールについてステップS34以降の処理を実行するための判断処理である。

【0116】ステップS34では、モジュールリストMからモジュールMiを取り出す。ステップS35は、モジュールMiについてのインタフェース定義に記述されている各ポートについてステップS36～S41を実行するための処理である。ステップS36では、インタフェース定義からポートPを抽出する。そして、そのポートPのビット幅が最小ビット幅パラメータWb以上か否かを調べる。ポートPのビット幅が最小ビット幅パラメータWb以上であればステップS37へ進み、そうでない場合は、ステップS35へ戻って次のポートを抽出する。

【0117】ステップS37では、ポートPが入力ポートであるか否かを調べる。ポートPが入力ポートであった場合には、ステップS40において、ポートPを入力ポートリストSiに登録する。また、ステップS38では、ポートPが出力ポートであるか否かを調べる。ポートPが出力ポートであった場合には、ステップS41において、ポートPを出力ポートリストSoに登録する。ポートPが入力ポートまたは出力ポートのいずれでもなかった場合には、ステップS39において、ポートPを入出力ポートリストSioに登録する。

【0118】上記処理により、ハードウェア仕様2において定義されている各ポートが、入力ポートリストSi、出力ポートリストSo、または入出力ポートリストSioに登録される。

【0119】ステップS51は、モジュールMiのモジュール宣言に記述されている各変数VについてステップS52～S55を実行するための処理である。ステップS52では、変数Vのビット幅が最小ビット幅パラメータWb以上であるか否かを調べる。変数Vのビット幅が最小ビット幅パラメータWb以上であればステップS53へ進み、そうでない場合はステップS51に戻って次の変数を抽出する。

【0120】ステップS53では、変数Vがレジスタ変数であるか否かを調べる。変数Vがレジスタ変数であれ

ば、ステップS55においてその変数Vをレジスタ変数リストSrに登録し、そうでない場合は、ステップS54においてその変数Vをバスワイヤ変数リストSwに登録する。

【0121】上記処理により、ハードウェア仕様2において定義されている各変数が、レジスタ変数リストSrまたはバスワイヤ変数リストSwに登録される。ステップS56は、モジュールMiの各モジュールインスタンスエイションについてステップS57およびS58を実行するための処理である。ステップS57では、モジュールインスタンスエイションm-jをモジュールリストMiに登録する。ステップS58では、モジュールインスタンスエイションm-jのポートをモジュールMi内の先に識別されている資源に関連づける。

【0122】ステップS61は、モジュールMi内の各同時処理CPについてステップS62～S72を実行するための処理である。ステップS62では、同時処理CPがcontinuous assignmentタイプであるか否かを調べる。ここで、同時処理CPがcontinuous assignmentタイプであれば、ステップS71において、その処理CPのデータ転送処理タイプとして「C (Concurrent)」を設定する。そして、ステップS72において、データ転送エッジを生成する。

【0123】ステップS63では、同時処理CPがguarded continuous assignmentタイプであるか否かを調べる。ステップS64では、同時処理CPが、combinationalalways blockタイプであるか否かを調べる。そして、ステップS63またはS64の判断結果が「Yes」ならば、ステップS70において、その処理CPのデータ転送処理タイプとして「C」を設定する。一方、ステップS63およびS64の判断結果が共に「No」ならば、ステップS65において、その処理CPのデータ転送処理タイプとして「S (Sequential)」を設定する。

【0124】ステップS66では、すべてのガード条件を認識する。ここで、各ガード条件は、それぞれ異なるデータ転送を制御する。ステップS67では、各ガード条件毎に、ガード表現およびガード信号を格納する。このとき、このガード条件により制御されるデータ転送の転送元および転送先を認識する。そして、ステップS68において、データ転送エッジを生成する。ステップS69は、各データ転送についてステップS66～S68を実行するための処理である。

【0125】上記処理により、ハードウェア仕様2において定義されている各データ転送に対応するデータ転送エッジが生成される。図35は、ステップS68またはS72のデータ転送エッジを生成する処理の詳細フローチャートである。

【0126】ステップS81では、データ転送の転送元が代入式であるか否かを調べる。転送元が代入式である

場合は、ステップS82において、その代入式に対応する擬似資源Sdtを生成する。ステップS83では、データ転送の転送元が定数であるか否かを調べる。転送元が定数である場合は、ステップS84において、その定数に対応する擬似資源Sdcを生成する。

【0127】データ転送の転送元が、代入式または定数のいずれでもない場合は、ステップS85において、転送元資源Sourceを認識する。ステップS86では、データ転送の転送元が、入力ポートリストSi、入出力ポートリストSio、バスワイヤ変数リストSwまたはレジスタ変数リストSrに属しているか否かを調べる。

【0128】擬似資源Sdtまたは擬似資源Sdcが生成された場合、あるいはステップS86の判断結果が「Yes」であった場合には、ステップS87において、転送先資源Sinkを認識する。ステップS88では、データ転送の転送先が、出力ポートリストSo、入出力ポートリストSio、バスワイヤ変数リストSwまたはレジスタ変数リストSrに属しているか否かを調べる。

【0129】ステップS88の判断結果が「Yes」であった場合には、ステップS89において、データ転送の転送元および転送先に基づいてエッジEを生成する。ステップS90では、データ転送エッジリストEdtに生成したエッジEを追加する。ステップS91では、エッジEをデータ転送の転送元及び転送先に関連づける。なお、この実施例では、データ転送の転送元または転送先がポート、レジスタ、バスワイヤ、または擬似資源のいずれでもなかった場合は、エッジEは生成されない。

【0130】上記処理により、ハードウェア仕様2において定義されている各データ転送に対して、そのデータ転送の転送元および転送先が定義されたデータ転送エッジが生成される。

【0131】図36および図37は、「非フラグメント等価」を検出する処理のフローチャートである。このフローチャートは、図4に示したフローチャートのステップS11の実施例である。なお、ここでは、入出力ポートについて説明するが、入力ポートおよび出力ポートについても同様の処理が実行される。

【0132】ステップS101では、入出力ポートリストSioをスキャンする。ステップS102は、入出力ポートリストSioに属する各要素（入出力ポート）についてステップS103以降の処理を実行するための処理である。ステップS103では入出力ポートリストSioから要素Sio-kを抽出する。

【0133】ステップS104では、抽出した要素Sio-kを待ち行列Qに加える。ステップS105では、待ち行列Qが空か否かを調べる。待ち行列Qが空のときはステップS102に戻り、待ち行列Qに1以上の要素が保持されている場合は、ステップS106において、待ち行列Qから要素qを抽出する。

【0134】ステップS107では、要素qが「子供資

源」を有するか否かを調べる。ここで、要素qの「子供資源」とは、例えば、要素qが設けられているモジュールの階層の下位の階層のモジュール内に設けられている資源であって、要素qに対応付けられているものをいう。対応関係は、ハードウェア仕様などに記述されている。

【0135】要素qが「子供資源」を有していなければ、ステップS114において待ち行列Qから要素qを削除した後にステップS102に戻り、要素qが「子供資源」を有している場合にはステップS108へ進む。ステップS108では、要素Sio-kに対応する非フラグメント等価クラスHekを生成する。そして、ステップS109において、Eqvl フラグをリセットする。

【0136】ステップS110は、要素qが有する各「子供資源」についてステップS121～S126を実行するための処理である。ステップS121では、要素qから「子供資源qc」を抽出する。ステップS122では、要素qのビット幅と「子供資源qc」のビット幅が互いに同じであるか否かを調べる。これらのビット幅が互いに異なる場合は、ステップS110に戻って次の「子供資源」を抽出する。上記ビット幅が互いに同じ場合、ステップS123においてEqvl フラグに「1」を設定する。ステップS124では、「子供資源qc」を待ち行列Qの要素として追加する。ステップS125では、「子供資源qc」を非フラグメント等価クラスHekの要素として追加する。ステップS126では、非フラグメント等価クラスHekと「子供資源qc」とを関連づける。

【0137】全ての「子供資源」についてステップS121～S126の処理が実行されると、ステップS111において、Eqvl フラグが「1」か否かを調べる。Eqvl フラグが「1」であれば、ステップS112において要素qを非フラグメント等価クラスHekに追加し、さらに、ステップS113において非フラグメント等価クラスHekと要素qとを関連づける。なお、Eqvl フラグが「1」でなかった場合には、ステップS112およびS113の処理はスキップされる。

【0138】このように、あるポートのビット幅とそのポートの「子供資源」のビット幅が互いに同じである場合、そのポートと「子供資源」は、「非フラグメント等価」とみなされる。

【0139】図38～図41は、「フラグメント等価」を検出する処理のフローチャートである。このフローチャートは、図4に示したフローチャートのステップS12の実施例である。なお、ここでは、入出力ポートについて説明するが、入力ポートおよび出力ポートについても同様の処理が実行される。

【0140】ステップS131では、入出力ポートリストSioをスキャンする。ステップS132は、入出力ポートリストSioに属する各要素（入出力ポート）についてステップS133以降を実行するための処理である。

ステップS133では、入出力ポートリストSioから要素Sio-lを抽出する。

【0141】ステップS134では、要素Sio-lがいずれかの等価クラスHekに属しているか否かを調べる。要素Sio-lがいずれかの等価クラスHekに属している場合は、ステップS135に進み、そうでない場合には、ステップS141～S146の処理を実行した後にステップS132に戻る。

【0142】ステップS141では、要素Sio-lが「子供資源」を有するか否かを調べる。要素Sio-lが「子供資源」を有する場合は、ステップS142以降の処理を実行し、そうでない場合は、ステップS132に戻る。ステップS142では、要素Sio-lに対応するフラグメント等価クラスOIFHelを生成する。ステップS143では、要素Sio-lをルート（出発点）として、要素Sio-lのビット幅よりも小さいビット幅を持つ「子供資源」を認識する。ステップS144では、各「子供資源」のためのフラグメント等価フィールドに「フラグメント等価クラスOIFHel」を設定する。ステップS145では、各「子供資源」のための非フラグメント等価フィールドに「非フラグメント等価クラスHek」を設定する。ステップS146では、各「子供資源」をフラグメント等価クラスOIFHelの要素として追加する。

【0143】要素Sio-lがいずれかの等価クラスHekに属している場合は、ステップS135において、フラグメントchild-found フラグをリセットする。ステップS136では、要素Sio-lをルートとして、要素Sio-lのビット幅よりも小さいビット幅を持った「子供資源」または要素Sio-lのビット幅よりも大きいビット幅を持った「子供資源」を探す。このとき、抽出された「フラグメント子供」の集合を“F-child”とする。一方、より大きなビット幅を持った「非フラグメント子供資源」の集合を“UG-child”とする。

【0144】ステップS137では、集合F-child の要素が存在するか否かを調べる。集合F-child の要素が存在するのであれば、ステップS138においてフラグメントchild-found フラグをセットする。ステップS139では、要素Sio-lに対応するフラグメント等価クラスOIFHelを生成する。

【0145】ステップS151では、集合F-child をスキャンする。ステップS152は、集合F-child に属する各要素についてステップS153～S156を実行するための処理である。ステップS153では、集合F-child から要素Cd-i を抽出する。ステップS154では、要素Cd-i の「親資源P-Cd-i」を抽出する。ステップS155では、要素Cd-i および親資源P-Cd-i のためのフラグメント等価フィールドに「フラグメント等価クラスOIFHel」を設定する。ステップS156では、要素Cd-i をフラグメント等価クラスOIFHelに加える。

【0146】集合F-childの要素が存在しない場合は、ステップS161～S172の処理が実行される。ステップS161～S165の処理は、基本的に、ステップS137、およびステップS151～S154の処理と同じである。ただし、ステップS161～S165では、集合UG-childから要素Cd-iが抽出され、さらに、その「親資源Prnt-i」が抽出される。

【0147】ステップS166では、要素Cd-iに対応するフラグメント等価クラスOIFHelを生成する。ステップS167は「親資源Prnt-i」の各要素について 10 ステップS168～S172を実行するための処理である。ステップS168では、「親資源Prnt-i」に属する要素Pi-kを抽出する。ステップS169では、要素Pi-kが属している非フラグメント等価クラスHekを探す。ステップS170では、非フラグメント等価クラスHekの各要素をフラグメント等価クラスOIFHelに加える。ステップS171では、非フラグメント等価クラスHekの各要素のためのフラグメント等価フィールドに「フラグメント等価クラスOIFHel」を設定する。そして、ステップS172において、各要素のための非 20 フラグメント等価フィールドから非フラグメント等価クラスHekを削除する。

【0148】図42は、「名称等価」を検出する処理のフローチャートである。このフローチャートは、図4に示したフローチャートのステップS13の実施例である。ステップS181では、レジスタリストSrをスキャンする。ステップS182は、レジスタリストSrの各要素についてステップS183～S189の処理を実行するための処理である。ステップS183では、レジ 30 スタリストSrから要素（レジスタ）Sr-lを抽出する。

【0149】ステップS184～S186では、抽出した要素Sr-lの名称と、出力ポートリストSoの各要素の名称とを比較する。そして、出力ポートリストSoの中に要素Sr-lの名称と同じ名称を持つ要素So-jが存在している場合には、ステップS187において、その要素So-jが属する非フラグメント等価クラスHekを探す。ステップS188では、要素Sr-lのための非フラ 40 グメント等価クラスフィールドに「非フラグメント等価クラスHek」を設定する。そして、ステップS189において、要素Sr-lを非フラグメント等価クラスHekの要素として追加する。

【0150】上記処理により、あるレジスタとそのレジスタと同じ名称が付与されている出力ポートとが、「名称等価」とみなされる。図43は、「フラグメントエッジ」を検出する処理のフローチャートである。この処理では、各データ転送エッジが「フラグメントエッジ」または「非フラグメントエッジ」のいずれに属するのかを判断するために有用な情報を生成する。なお、このフロー 50 チャートは、図4に示したフローチャートのステップ

S14の実施例である。

【0151】ステップS191では、データ転送エッジリストEdtをスキャンする。ステップS192は、データ転送エッジリストEdtの各要素についてステップS193～S200を実行するための処理である。ステップS193では、データ転送エッジリストEdtから要素（エッジ）Ejを抽出する。このとき、エッジEjに対応するデータ転送の転送元および転送先を認識する。

【0152】ステップS194では、エッジEjに対応するデータ転送の転送先が「フラグメント等価」である 10 かどうかを調べる。また、その転送先が「フラグメント等価」であれば、ステップS195において、エッジEjに対応するデータ転送の転送元が「フラグメント等価」であるかどうかを調べる。そして、その転送先および転送元が共に「フラグメント等価」であった場合には、ステップS196において両フラグメントフラグをセットし、転送先のみが「フラグメント等価」であった場合には、ステップS197において転送先フラグメントフラグをセットする。一方、転送先が「フラグメント等価」 20 でなかった場合には、ステップS198において、エッジEjに対応するデータ転送の転送元が「フラグメント等価」であるかどうかを調べる。そして、転送元のみが「フラグメント等価」であった場合には、ステップS199において、転送元フラグメントフラグをセットし、転送先および転送元がいずれも「フラグメント等価」でなかった場合には、ステップS200において非フラグメントフラグをセットする。

【0153】上記処理により、各データ転送に対応するエッジが、両フラグメントエッジ、転送先フラグメントエッジ、転送元フラグメントエッジ、または非フラグメントエッジに分類される。

【0154】図44～図48は、「等価エッジ」を検出する処理のフローチャートである。このフローチャートは、図4に示したフローチャートのステップS15の実 30 施例である。

【0155】ステップS211では、データ転送エッジリストEdtをスキャンする。ステップS212は、データ転送エッジリストEdtの各要素についてステップS213以降の処理を実行するための処理である。ステップS213では、データ転送エッジリストEdtから要素（エッジ）Ejを抽出する。このとき、エッジEjに対 40 応するデータ転送の転送元および転送先を認識する。

【0156】ステップS214では、エッジEjが転送元フラグメントかどうかを調べる。エッジEjが転送元フラグメントであればステップS215へ進み、そうでない場合は、ステップS241へ進む。ステップS215では、転送元のフラグメント等価を「FHej」とする。ステップS216では、フラグメント等価FHejに属する要素をスキャンする。ステップS217は、フラグメン 50 ト等価FHejに属する各要素についてステップS218

～S224を実行するための処理である。

【0157】ステップS218では、フラグメント等価FHejから要素（ポート等）Niを抽出する。ステップS219では、要素Niを転送元ノードまたは転送先ノードとして含むすべてのエッジを抽出する。ステップS220では、抽出したエッジのリストをスキャンする。ステップS221は、ステップS219で抽出した各エッジについてステップS222～S224を実行するための処理である。ステップS222では、上記抽出したエッジのリストから要素（エッジ）Ekを抽出する。このとき、エッジEkに対応するデータ転送の転送先ノードを「Head-k」とする。ステップS223では、この転送先ノードHead-kが、ステップS213で抽出した転送先と同じであるか否かを調べる。そして、それらが互いに同じ場合は、ステップS224において、エッジEjの等価エッジの集合Eqv-Tail-EjにエッジEkを追加する。

【0158】フラグメント等価FHejに属するすべての要素についてステップS218～S224の処理を実行すると、ステップS231において、集合Eqv-Tail-Ejに属するエッジのノードのビット幅ベクトルに基づいてインターバルグラフIGを作成する。

【0159】ここで、図49～図51を参照しながら、インターバルグラフについて説明する。ここでは、図49に示すデータ転送グラフが生成されているものとする。また、図49に示すノード間のデータ転送は、図50に示すように定義されているものとする。そして、各エッジ毎に、ガード条件に対応する制御信号、転送先ノードにおいて使用されるビット、および転送先ノードにおいてビットされるビットが定義されているものとする。この定義によれば、例えば、エッジRTは、制御信号G3が与えられたときに、ノードRの第7～0ビットがノードTの第7～0ビットへ転送されることを表している。なお、各データ転送は、T1～T10を用いて表わされている。

【0160】データ転送グラフからインターバルグラフを作成する場合、まず、各データ転送を点（ドット）で表す。そして、データ転送毎に、転送先ノードにおいて使用すべきビットがオーバーラップするデータ転送を抽出し、それらを接続する。

【0161】例えば、データ転送T1に注目する。データ転送T1によるデータは、ノードTの第7～0ビットに書き込まれる。このとき、データ転送T2によるデータはノードTの第3～0ビットに書き込まれる。したがって、データ転送T1およびデータ転送T2により転送されるデータは、ノードTにおいてオーバーラップすることになる。同様に、データ転送T3、T8、T9及びT10によるデータも、ノードTにおいてデータ転送T1によるデータとオーバーラップする。この場合、データ転送T1を表す点は、図51に示すように、データ転送T2

、T3、T8、T9およびT10を表す点と接続される。

【0162】この後、他のデータ転送についても同様の方法を実行することにより、図51に示すインターバルグラフが得られる。フローチャートの説明に戻る。ステップS232では、インターバルグラフIGにおいて、部品（図51における「点」）同士の接続を調べる。ステップS233では、非単体部品の集合NSCCを抽出する。各集合NSCCには、インターバルグラフにおいて互いにメッシュ状に接続された点に対応する複数のデータ転送から構成される。例えば、図51において、点T1、T2、T3、T10は、互いにメッシュ状に接続されており、これらの点にそれぞれ対応するデータ転送は1つの集合NSCCに属する。ステップS234では、単体部品の集合SCCを抽出する。集合SCCには、いずれの集合NSCCにも属していないデータ転送が属することになる。

【0163】ステップS235では、各集合NSCCに属し、且つ集合Eqv-Tail-Ejに含まれるエッジから独立サブセットInd-Eqv-Tail-Ejを作成する。ステップS236では、集合Eqv-Tail-Ejから独立サブセットInd-Eqv-Tail-Ejを削除する。ステップS237では、集合NSCCに基づいて、集合Eqv-Tail-Ejを1以上のサブグループに分割する。そして、ステップS238において、各サブグループごとにエッジリストEfinalを得る。そして、ステップS239において、各サブグループの集合Eqv-Tail-EjをエッジリストEfinalに追加する。

【0164】エッジEjが転送元フラグメントでなかった場合（S214：No）は、ステップS241において、エッジEjが転送先フラグメントまたは両フラグメントであるか否かを調べる。エッジEjが転送先フラグメントまたは両フラグメントであればステップS242へ進み、そうでない場合は、ステップS249においてそのエッジEjを単体ブロックとしてエッジリストEfinalに加えた後にステップS212に戻る。

【0165】ステップS242では、非フラグメントの親資源UF-Parent-jが見つかるまでエッジEjに対応するデータ転送の転送先の親資源フィールドをトラバースしていく。ステップS243では、親資源UF-Parent-jが属する非フラグメント等価クラスを「UHPej」とする。ステップS244では、非フラグメント等価クラスUHPejをスキャンする。ステップS245は、非フラグメント等価クラスUHPejに属する各要素についてステップS246～S248を実行するための処理である。ステップS246では、非フラグメント等価クラスUHPejから要素（ノード）Niを抽出する。ステップS247では、ノードNiを転送先ノードとして含むすべてのエッジを抽出する。ステップS248では、抽出された各エッジを等価エッジの集合Eqv-Head-Ejに追加

する。

【0166】ステップS251では、非フラグメント等価クラスUHPejから抽出されたノードNiがフラグメント等価か否かを調べる。ノードNiがフラグメント等価であればステップS252へ進み、そうでなければステップS245に戻る。ステップS252では、ノードNiのフラグメント等価クラスを「FHej」とする。ステップS253では、フラグメント等価クラスFHejをスキャンする。ステップS254は、フラグメント等価クラスFHejに属する各要素についてステップS255

～S257を実行するための処理である。ステップS255では、フラグメント等価クラスFHejからノードNkを抽出する。ステップS256では、転送先ノードとしてノードNkを含むすべてのエッジを検出する。そして、ステップS257において、抽出した各エッジを等価エッジの集合E-Head-Ejに追加する。

【0167】各ノードについてステップS246～S248を実行すると（ステップS245：Yes）、続いてステップS261～S268を実行する。ステップS261～S268は、基本的に、ステップS231～S237、およびS239と同じである。ただし、ステップS261～S268では、集合Eqv-Head-Ejに属するエッジに係わるインターバルグラフIGが作成され、そのグラフを利用してエッジリストEfinalが得られる。

【0168】図52～図55は、「レジスタファイル」を検出する処理のフローチャートである。このフローチャートは、図4に示したフローチャートのステップS16の実施例である。

【0169】ステップS271では、データ転送エッジリストEdtをスキャンする。ステップS272は、データ転送エッジリストEdtに属する各要素についてステップS273～S284を実行するための処理である。ステップS273では、データ転送エッジリストEdtから要素（エッジ）Ejを抽出する。また、エッジEjに対応するデータ転送の転送元Sourceおよび転送先Sinkを検出する。

【0170】ステップS274では、エッジEjの転送元または転送先がレジスタであるか否かを調べる。転送元または転送先がレジスタであれば、ステップS275へ進み、そうでない場合はステップS272に戻る。ステップS275では、エッジEjのガード表現のサポートセットを構成する制御信号の集合SSS-Ejを作成する。ステップS276では、検出した転送先がレジスタであるか否かを調べる。転送先がレジスタであればステップS277へ進み、そうでない場合はステップS277～S280をスキップする。

【0171】ステップS277では、制御信号の集合SSS-Ejを転送先に関連づける。ステップS278では、転送先がレジスタであるデータ転送のガード表現に含まれる制御信号の集合Sink-Support-Setを得る。ステ

ップS279では、制御信号の集合SSS-Ejを制御信号の集合Sink-Support-Setに関連づける。ステップS280では、転送先を転送先レジスタの集合Sink-Registersに追加する。

【0172】ステップS281では、検出した転送元がレジスタであるか否かを調べる。転送元がレジスタであればステップS282へ進み、そうでない場合は、ステップS272に戻る。ステップS282では、制御信号の集合SSS-Ejを転送元に関連づける。ステップS283では、制御信号の集合SSS-Ejを制御信号の集合Source-Support-Setに関連づける。そして、ステップS284において、転送元を転送元レジスタの集合Source-Registersに追加する。

【0173】すべてのエッジについてステップS273～S284の処理を実行すると、ステップS291において、制御信号の集合Source-Support-Setの結合体U-Source-Setsを作成する。また、ステップS292において、制御信号の集合Sink-Support-Setの結合体U-Sink-Setsを作成する。ステップS293では、結合体U-Source-Setsに属するユニークな要素から構成されるランダムトータルオーダTo-U-Source-Setsを作成する。また、ステップS294では、結合体U-Sink-Setsに属するユニークな要素から構成されるランダムトータルオーダTo-U-Sink-Setsを作成する。

【0174】ステップS295では、Sink-Support-Setをスキャンする。ステップS296は、Sink-Support-Setに属するすべてのセットについてステップS297～S300を実行するための処理である。ステップS297では、Sink-Support-SetからSSS-Ejを抽出する。ステップS298では、To-U-Sink-Setsに基づいてSSS-Ejのビットベクトル表現BV-SSS-Ejを作成する。ここで、SSS-Ejの要素に対応するエントリに「1」を設定し、他の要素に「0」を設定する。ステップS299では、BV-SSS-EjをSSS-Ejに関連づける。そして、ステップS300において、BV-SSS-EjをSSS-Ejに追加する。このとき、BV-Sink-Support-Setsは、Sink-Support-Setのビットベクトル表現の集合である。

【0175】すべてのセットについてステップS297～S300の処理を実行すると、ステップS311へ進む。ステップS311は、Source-Support-Setに属するすべての集合についてステップS312～S315を実行するための処理である。ステップS312～S315は、基本的に上述したステップS297～S300と同じである。

【0176】ステップS316では、BV-Sink-Support-Setのベクトルリストをスキャンする。ステップS317は、ベクトルリストに属する各ベクトルについてステップS318～S324を実行するための処理である。ステップS318では、ベクトルリストからBV-

10

20

30

40

50

SSS-Ej を抽出する。ステップS319では、BV-Sink-Support-Set のベクトルリストを再スキャンする。ステップS320は、ベクトルリストに属する各ベクトルについてステップS321~S323を実行するための処理である。ステップS321では、ベクトルリストからBV-SSS-Ek を抽出する。ステップS322では、BV-SSS-Ej とBV-SSS-Ek が同じであるか否かを調べる。これらが互いに同じであれば、ステップS323において、BV-SSS-Ek に対応する転送先レジスタReg-sink-kを検出する。そして、すべてのベクトルについてステップS321~S323の処理を実行すると、ステップS324において、転送先レジスタReg-sink-kを転送先レジスタファイルの集合Sink-Reg-Filesに追加する。

【0177】ステップS331~S339は、基本的にステップS316~S324の処理と同じである。ただし、ステップS331~S339では、BV-Sink-Support-Set のベクトルリストに属する各要素について処理が行われ、所定の転送元レジスタが転送元レジスタファイルの集合Source-Reg-Filesに追加される。

【0178】ステップS341では、転送先レジスタファイルの集合Sink-Reg-Filesのリストをスキャンする。ステップS342は、そのリストに属する各集合についてステップS343~S348を実行するための処理に相当する。ステップS343では、上記のリストから集合Sink-Reg-File-k を抽出する。ステップS345では、すべての集合が処理されたか否かを調べる。未処理のセットがあれば、ステップS346において、集合Source-Reg-File-l を抽出する。ステップS347では、集合Sink-Reg-File-k と集合Source-Reg-File-l とが同一であるか否かを調べる。そして、それらが互いに同じであれば、ステップS348において、集合Source-Reg-File-k を集合Reg-Files に追加する。上記処理により、レジスタファイルに属するレジスタが得られる。

【0179】このように、図32~図35に示すフローチャートに処理により生成されたデータ転送グラフは、上述のフローチャートの処理により最適化される。そして、その最適化の結果は、エッジリストEfinal である。

【0180】図57は、最適化されたデータ転送グラフから隣接ノードリストを作成する処理のフローチャートである。なお、このフローチャートは、図19に示したフローチャートのステップS21の実施例である。

【0181】ステップS351では、エッジリストEfinal をスキャンする。ステップS352は、エッジリストEfinal に属する各サブグループについてステップS353~S361を実行するための処理に相当する。ステップS353では、エッジリストEfinal からサブグループPi を抽出する。

【0182】ステップS354は、サブグループPi に

属する各エッジについてステップS355~S361の処理を実行するための処理に相当する。ステップS355では、サブグループPi からエッジEj を抽出する。このとき、エッジEj の転送先ノードおよび転送元ノードが属する非フラグメント等価クラスにそれぞれ対応する等価クラスHead-Hejおよび等価クラスTail-Hejを抽出する。ステップS356では、非フラグメント等価クラスHead-Hejが隣接ノードリストのノードセットNの要素であるか否かを調べる。そして、非フラグメント等価クラスHead-HejがノードセットNの要素でなかった場合には、ステップS357においてそれをノードセットNに追加する。同様に、ステップS358では、非フラグメント等価クラスTail-HejがノードセットNの要素であるか否かを調べる。そして、非フラグメント等価クラスTail-HejがノードセットNの要素でなかった場合には、ステップS359においてそれをノードセットNに追加する。

【0183】ステップS360では、隣接ノードリストのノードセットNにおける非フラグメント等価クラスHead-Hejの隣接リストに非フラグメント等価クラスTail-Hejを追加する。また、非フラグメント等価クラスHead-Hejの入力エッジ数をインクリメントする。一方、ステップS361では、ノードセットNにおける非フラグメント等価クラスTail-Hejの隣接リストに非フラグメント等価クラスHead-Hejを追加する。そして、非フラグメント等価クラスTail-Hejの入力エッジ数をインクリメントする。

【0184】図58~図60は、検証ツールに入力すべきプロパティスクリプトを生成する処理のフローチャートである。なお、このフローチャートは、図19に示したフローチャートのステップS22の実施例である。

【0185】ステップS371では、隣接ノードリストのノードセットNをスキャンする。ステップS372は、ノードセットNに属する各ノードについてステップS373以降の処理を実行するための処理である。ステップS373では、ノードセットからノードNj を抽出する。

【0186】ステップS374では、ノードNj のファンインエッジ数が1以上であるか否かを調べる。ファンインエッジ数が1以上であればステップS375へ進み、そうでない場合はステップS372に戻る。ステップS375では、転送先ノードがノードNj であるエッジ(ノードNj へのファンインエッジ)の集合Di-Arcs-into-Nj を得る。ステップS376では、ノードNj のファンアウトエッジ数が1以上であるか否かを調べる。出力エッジ数が1以上であればステップS377へ進み、そうでない場合はステップS377をスキップする。

【0187】ステップS377では、転送元ノードがノードNj であるエッジ(ノードNj からのファンアウト

エッジ)の集合Di-Arcs-out-of-Njを得る。ステップS378では、ノードNjがレジスタであるか否かを調べる。ノードNjがレジスタであればステップS379へ進み、そうでない場合はステップS411へ進む。ステップS379では、集合Di-Arcs-into-Njに属するファンインエッジが存在するか否かを調べる。集合Di-Arcs-into-Njに属するファンインエッジが存在する場合はステップS380において、それらのファンインエッジ同士の資源競合プロパティを生成する。そうでない場合はステップS372に戻る。

【0188】ステップS381では、集合Di-Arcs-out-of-Njに属するファンアウトエッジが存在するか否かを調べる。集合Di-Arcs-out-of-Njに属するファンアウトエッジが存在する場合は、ステップS380において、集合Di-Arcs-into-Njと集合Di-Arcs-out-of-Njとの積PS1を生成する。PS1を「Em、En」と表す。なお、「Em」及び「En」は、それぞれ集合Di-Arcs-into-Njおよび集合Di-Arcs-out-of-Njの要素である。一方、集合Di-Arcs-out-of-Njに属するファンアウトエッジが存在しない場合は、ステップS372に戻る。

【0189】ステップS391では、PS1をスキャンする。ステップS392は、PS1の各要素についてステップS393～S395の処理を実行するための処理に相当する。ステップS393では、PS1から要素PS-k(Ea、Eb)を抽出する。ステップS394では、集合Di-Arcs-out-of-Njの要素Ebの処理タイプが順次転送であるか否かを調べる。そして、処理タイプが順次転送であれば、ステップS395において、クロック歪みによるエラーをチェックするための資源競合プロパティを生成する。

【0190】ステップS396では、集合Di-Arcs-into-Njと集合Di-Arcs-into-Njとの積PS2を生成する。続いて、ステップS397では、エッジEjのガード表現「Gi」としたときに、集合Di-Arcs-out-of-Njに属する各ファンアウトエッジEiに対して論理式No-Read=(OR(Gi))を生成する。ステップS398では、PS2をスキャンする。ステップS399は、PS2に属する各要素について処理400およびS401を実行するための処理に相当する。

【0191】ステップS400では、PS2からPS2-k=(Ec、Ed)を抽出する。そして、ステップS401において、レジスタ漏れプロパティEF(Gi*EX(E(No-ReadUGm)))を生成する。

【0192】ノードNjがレジスタでなかった場合(ステップS378:No)は、ステップS411において、そのノードNjが出力ポートであるか否かを調べる。ノードNjが出力ポートであれば、ステップS412において、集合Di-Arcs-into-Njに要素が存在するか否かを調べる。すなわち、その出力ポートへのファンインエッジが存在するか否かを調べる。ファンインエッジ

がある場合には、ステップS413において、それらエッジ同士のすべての組合せに対して資源競合プロパティを生成する。

【0193】ステップS414では、ノードNjが入力ポートであるか否かを調べる。ノードNjが入力ポートであれば、ステップS415において、集合Di-Arcs-into-Njおよび集合Di-Arcs-into-Njの結合体Union-Arcs-of-Njを生成する。そして、ステップS416において、Union-Arcs-of-NjとUnion-Arcs-of-Njとの積PS3を生成する。

【0194】ステップS417では、PS3をスキャンする。ステップS418は、PS3に属する各要素についてステップS419～S421の処理を実行するための処理に相当する。ステップS419では、PS3からPS3-k=(Ee、Eg)を抽出する。ステップS420では、エッジEeとエッジEgが同じものであるか調べる。それらのエッジが同じであった場合は、ステップS421において、資源競合プロパティを生成する。

【0195】上述したプロパティを生成する機能は、コンピュータを用いて上述のフローチャートに示した処理を記述したプログラムを実行することにより実現される。そのプログラムを実行するコンピュータ100のブロック図を図61に示す。

【0196】CPU101は、上述のフローチャートに示した処理を記述したプログラムを記憶装置102からメモリ103にロードして実行する。記憶装置102は、たとえばハードディスクであり、上記プログラムを格納する。一方、メモリ103は、例えば半導体メモリであり、CPU101の作業領域として使用される。

【0197】記録媒体ドライバ104は、CPU101の指示に従って可搬性記録媒体105にアクセスする。可搬性記録媒体105は、例えば、半導体デバイス(PCカード等)、磁気的作用により情報が入出力される媒体(フロッピーディスク、磁気テープなど)、光学的作用により情報が入出力される媒体(光ディスクなど)を含む。通信制御装置106は、CPU101の指示に従って網との間でデータを送受信する。

【0198】図23は、本発明に係わるソフトウェアプログラムなどの提供方法を説明する図である。本発明に係わるプログラムは、例えば、以下の3つの方法の中の任意の方法により提供される。

【0199】(a) コンピュータ100にインストールされて提供される。この場合、プログラム等は、たとえば、出荷前にプレインストールされる。

(b) 可搬性記録媒体に格納されて提供される。この場合、可搬性記録媒体105に格納されているプログラム等は、基本的に、記録媒体ドライバ104を介して記憶装置102にインストールされる。

【0200】(c) 網上のサーバから提供される。この場合、基本的には、コンピュータ100がサーバに格納さ

10

20

30

40

50

れているプログラム等をダウンロードすることによってそのプログラム等を取得する。網は、無線網を含む。

【0201】なお、上述の実施例では、検証すべきプロパティとして、資源競合およびレジスタ漏れを採り上げているが、本発明が生成するプロパティはこれに限定されるものではない。

【0202】また、データ資源としてレジスタ、ポート、バスを採り上げているが、本発明が対象とする資源はこれに限定されるものではない。

【0203】

【発明の効果】本発明によれば、ハードウェア記述言語で記述された仕様から、そのハードウェアにおいて資源競合またはレジスタ漏れが発生するかどうかを検証するためのプロパティを自動的に生成できる。このため、ハードウェアの設計ミスを早い段階で修正でき、IC等の開発のための時間およびコストが節約される。また、デバッグ作業が減るので、設計者は、余った労力を他の作業に投入できる。

【図面の簡単な説明】

【図1】本発明の一実施形態のプロパティ作成ツールが仕様される環境を説明する図である。

【図2】プロパティ生成ツールの動作を説明するフローチャートである。

【図3】データ転送グラフの一例である。

【図4】最適化処理のフローチャートである。

【図5】ハードウェアの例である。

【図6】図5に示すハードウェアを記述する階層的に方法を説明する図である。

【図7】「資源の等価」を利用してデータ転送グラフを最適化する例を示す図である。

【図8】「分岐」を説明する図である。

【図9】レジスタファイルを説明する図である。

【図10】Verilog で記述されたハードウェア仕様の一例（その1）である。

【図11】Verilog で記述されたハードウェア仕様の一例（その2）である。

【図12】Verilog で記述されたハードウェア仕様の一例（その3）である。

【図13】Verilog で記述されたハードウェア仕様の一例（その4）である。

【図14】Verilog で記述されたハードウェア仕様の一例（その5）である。

【図15】Verilog で記述されたハードウェア仕様の一例（その6）である。

【図16】図10～図15に示したハードウェア仕様から生成されたデータ転送グラフである。

【図17】疑似資源を説明する図である。

【図18】最適化されたデータ転送グラフの例である。

【図19】プロパティ作成処理のフローチャートである。

【図20】隣接ノードリストの例である。

【図21】「資源競合」のプロパティの例である。

【図22】「レジスタ漏れ」のプロパティの例である。

【図23】クロックの歪みを考慮した場合の「資源競合」のプロパティの例である。

【図24】Verilog で記述されたハードウェア仕様の一例（その1）である。

【図25】Verilog で記述されたハードウェア仕様の一例（その2）である。

10 【図26】Verilog で記述されたハードウェア仕様の一例（その3）である。

【図27】図24～図26に示したハードウェア仕様から生成されたデータ転送グラフである。

【図28】図27に示すグラフにおけるデータ転送の条件を示す図である。

【図29】レジスタファイルを検出する処理を説明する図である。

【図30】最適化されたデータ転送グラフの例である。

【図31】最適化されたグラフにおいて使用される条件を示す図である。

【図32】データ転送グラフを生成する方法のフローチャート（その1）である。

【図33】データ転送グラフを生成する方法のフローチャート（その2）である。

【図34】データ転送グラフを生成する方法のフローチャート（その3）である。

【図35】データ転送エッジを生成する処理の詳細フローチャートである。

30 【図36】「非フラグメント等価」を検出する処理のフローチャート（その1）である。

【図37】「非フラグメント等価」を検出する処理のフローチャート（その2）である。

【図38】「フラグメント等価」を検出する処理のフローチャート（その1）である。

【図39】「フラグメント等価」を検出する処理のフローチャート（その2）である。

【図40】「フラグメント等価」を検出する処理のフローチャート（その3）である。

40 【図41】「フラグメント等価」を検出する処理のフローチャート（その4）である。

【図42】「名称等価」を検出する処理のフローチャートである。

【図43】「分岐エッジ」を検出する処理のフローチャートである。

【図44】「等価エッジ」を検出する処理のフローチャート（その1）である。

【図45】「等価エッジ」を検出する処理のフローチャート（その2）である。

50 【図46】「等価エッジ」を検出する処理のフローチャート（その3）である。

【図47】「等価エッジ」を検出する処理のフローチャート（その4）である。

【図48】「等価エッジ」を検出する処理のフローチャート（その5）である。

【図49】データ転送グラフの例である。

【図50】データ転送の定義である。

【図51】インターバルグラフの例である。

【図52】「レジスタファイル」を検出する処理のフローチャート（その1）である。

【図53】「レジスタファイル」を検出する処理のフローチャート（その2）である。

【図54】「レジスタファイル」を検出する処理のフローチャート（その3）である。

【図55】「レジスタファイル」を検出する処理のフローチャート（その4）である。

【図56】「レジスタファイル」を検出する処理のフローチャート（その5）である。

*

*【図57】隣接ノードリストを作成する処理のフローチャートである。

【図58】プロパティスクリプトを生成する処理のフローチャート（その1）である。

【図59】プロパティスクリプトを生成する処理のフローチャート（その2）である。

【図60】プロパティスクリプトを生成する処理のフローチャート（その3）である。

【図61】本発明の機能を記述したプログラムを実行するコンピュータのブロック図である。

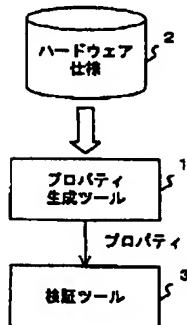
【図62】本発明に係わるソフトウェアプログラムなどの提供方法を説明する図である。

【符号の説明】

- 1 プロパティ生成ツール
- 2 ハードウェア仕様
- 3 検証ツール

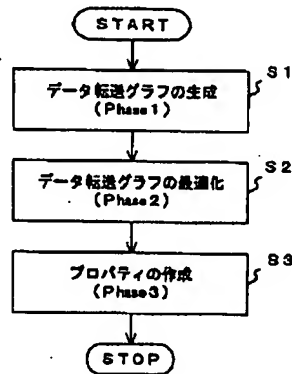
【図1】

本発明の一実施形態のプロパティ作成ツールが仕様される環境を説明する図



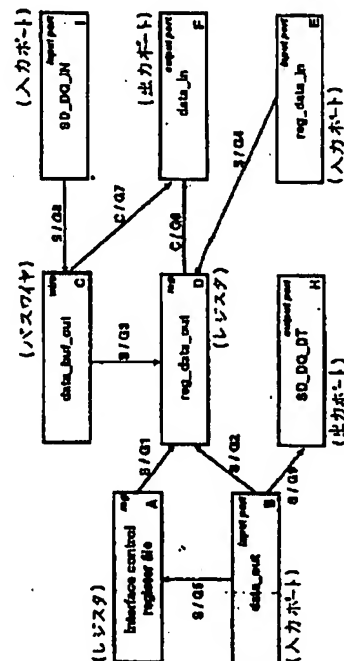
【図2】

プロパティ生成ツールの動作を説明するフローチャート



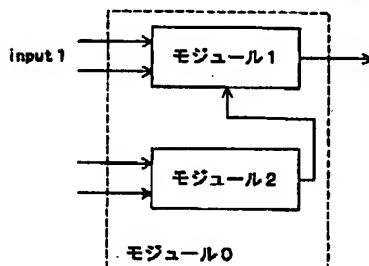
【図3】

データ転送グラフの一例



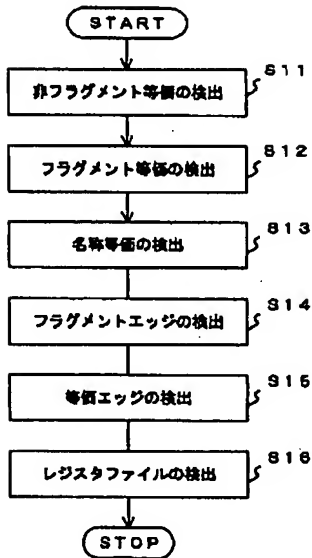
【図5】

ハードウェアの例

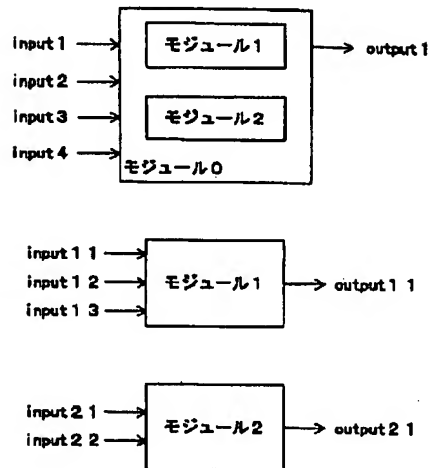


【図4】

最適化処理のフローチャート

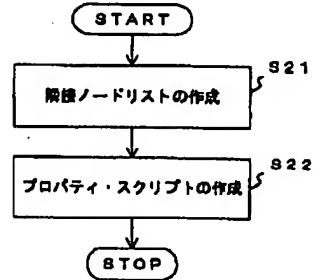


【図6】

図6に示すハードウェアを記述する
階層的に方法を説明する図

【図19】

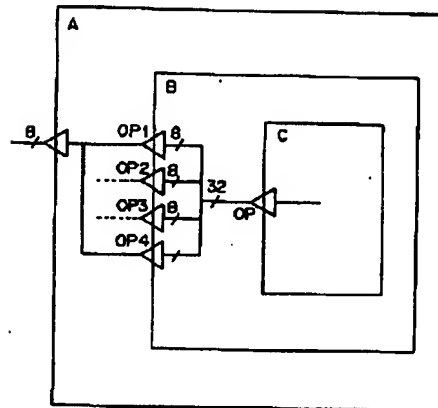
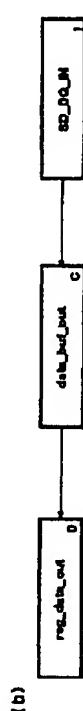
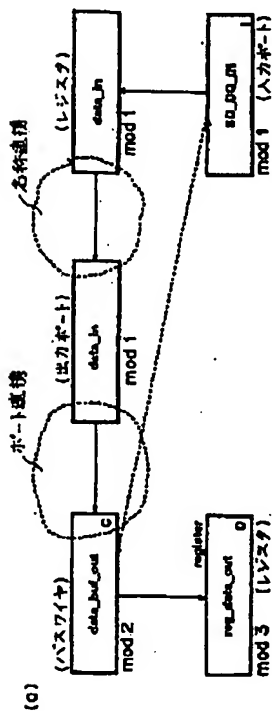
プロパティ作成処理のフローチャート



【図8】

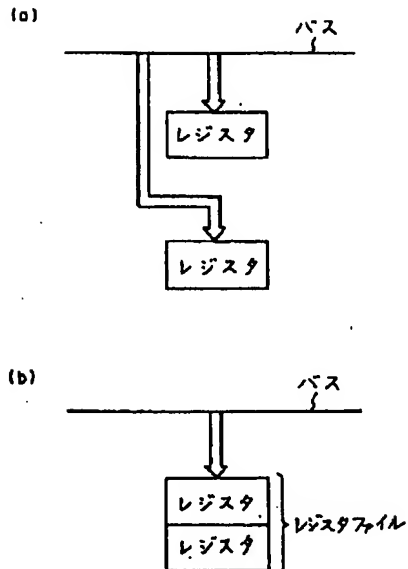
「分岐」を説明する図

【図7】

「資源の等価」を利用して
データ転送フラグを最適化する例を示す図

【図9】

レジスタファイルの説明図



【図10】

Verilogで記述されたハードウェアは様の一例(7の1)

```
module MOD 0 (GLOBAL_CLK, INPUT1, INPUT2, OUTPUT1, OUTPUT2,
              CTRL1, CTRL2, CTRL3, CTRL4, CTRL5, CTRL6, CTRL7);
```

```
input    GLOBAL_CLK;
input [15:0] INPUT1;
input [15:0] INPUT2;
output [15:0] OUTPUT1;
output [15:0] OUTPUT2;
input    CTRL1;
input    CTRL2;
input    CTRL3;
input    CTRL4;
input    CTRL5;
input    CTRL6;
input    CTRL7;

wire [15:0] CONEKT;
```

```
MOD1 MOD1_INST_A
(.GLOBAL_CLK(GLOBAL_CLK),
 .CTRL1(CTRL1),
 .CTRL2(CTRL2),
 .IPORT1(CONEKT[15:8]),
 .IPORT2(CONEKT[7:0]),
 .IPORT3(INPUT1[15:8]),
 .IPORT4(INPUT1[7:0]),
 .OPORT1(OUTPUT1[15:8]),
 .OPORT2(OUTPUT1[7:0])
);
```

```
MOD2 MOD2_INST_A
(.GLOBAL_CLK(GLOBAL_CLK),
 .CTRL3(CTRL3),
 .CTRL4(CTRL4),
 .CTRL5(CTRL5),
 .CTRL6(CTRL6),
 .CTRL7(CTRL7),
 .IPORT5(OUTPUT2),
 .OPORT3(OUTPUT2),
 .OPORT4(CONEKT)
);
```

```
endmodule
```

【図20】

隣接ノードリストの例

転送元ノード	転送先ノード
E	L
F	L
F	M
G	M
H	N
I	N
J	O
K	O
Q	V
L	V
M	V
R	V

【図11】

Verilogで記述されたハードウェア仕様の一例 (例2)

```

module MOD1 (GLOBAL_CLK, CTRL1, CTRL2, IPOR1, IPOR2,
             IPOR3, IPOR4, OPOR1, OPOR2);
    input GLOBAL_CLK;
    input CTRL1;
    input CTRL2;
    input [7:0] IPOR1;
    input [7:0] IPOR2;
    input [7:0] IPOR3;
    input [7:0] IPOR4;
    output [7:0] OPOR1;
    output [7:0] OPOR2;

    reg [7:0] REG_A;
    reg [7:0] REG_B;
    reg [7:0] REG_C;

    C1 : assign OPOR1 = REG_A;
    C2 : assign OPOR2 = REG_B;

```

【図12】

Verilogで記述されたハードウェア仕様の一例 (例3)

```

S1 : always @ (posedge (GLOBAL_CLK))
    begin
        if (CTRL1)
            begin
                REG_A = IPOR1;
                REG_B = IPOR3;
            end
    end

S2 : always @ (posedge (GLOBAL_CLK))
    begin
        if (CTRL2)
            begin
                REG_A = IPOR2;
                REG_B = IPOR4;
            end
    end
endmodule

```

【図13】

Verilogで記述されたハードウェア仕様の一例 (例4)

```

module MOD2 (GLOBAL_CLK, CTRL3, CTRL4, CTRL5,
             CTRL6, CTRL7, IPOR5, OPOR3, OPOR4);
    input GLOBAL_CLK;
    input CTRL3;
    input CTRL4;
    input CTRL5;
    input CTRL6;
    input CTRL7;
    input [15:0] IPOR5;
    output [15:0] OPOR3;
    output [15:0] OPOR4;

    reg [15:0] REG_C;
    reg [15:0] REG_D;
    reg [15:0] REG_E;
    reg [15:0] REG_F;

    C3 : assign OPOR3 = REG_E;
    C4 : assign OPOR4 = REG_F;

    initial
    begin
        REG_C = 16'b0000000000000000;
        REG_D = 16'b0000000000000001;
        REG_E = 16'b0000000000000000;
        REG_F = 16'b0000000000000000;
    end

```

【図14】

Verilogで記述されたハードウェア仕様の一例 (例5)

```

S3 : always @ (posedge (GLOBAL_CLK))
    begin
        if (CTRL3)
            begin
                REG_C = IPOR5;
            end
        else
            begin
                REG_D = IPOR5;
            end
    end

S4 : always @ (posedge (GLOBAL_CLK))
    begin
        if (CTRL4)
            begin
                REG_C = REG_C + REG_D;
                REG_D = REG_C / REG_D;
            end
    end

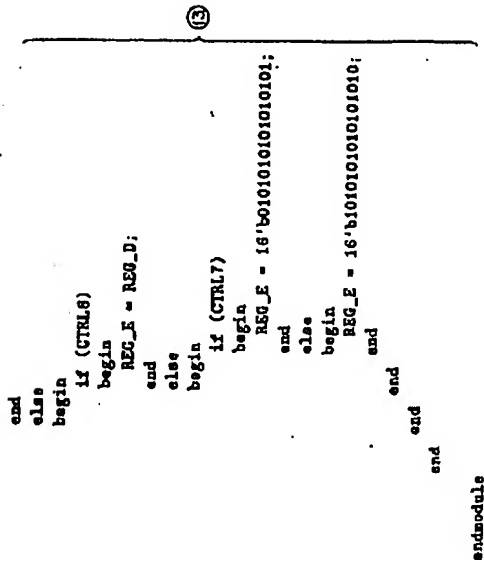
S5 : always @ (posedge (GLOBAL_CLK))
    begin
        REG_F = REG_E - REG_D;
    end

S6 : always @ (posedge (GLOBAL_CLK))
    begin
        if (CTRL5)
            begin
                REG_E = REG_C;
            end
    end

```

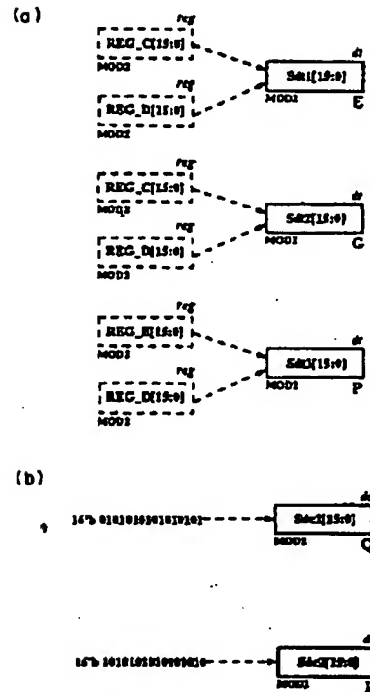
【図15】

Verilogで記述されたハードウェアは様の一例 (その5)



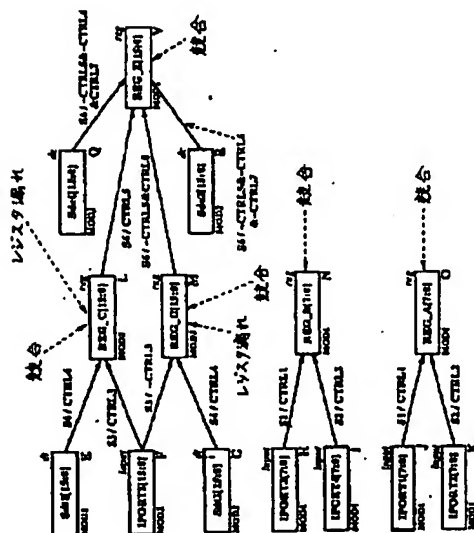
【図17】

疑似資源を説明する図



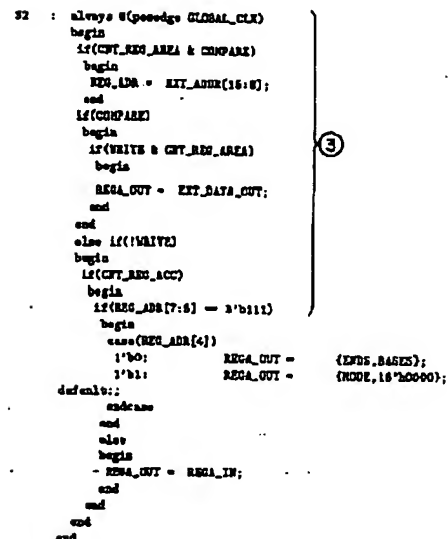
【図18】

最適化されたデータ転送グラフの例



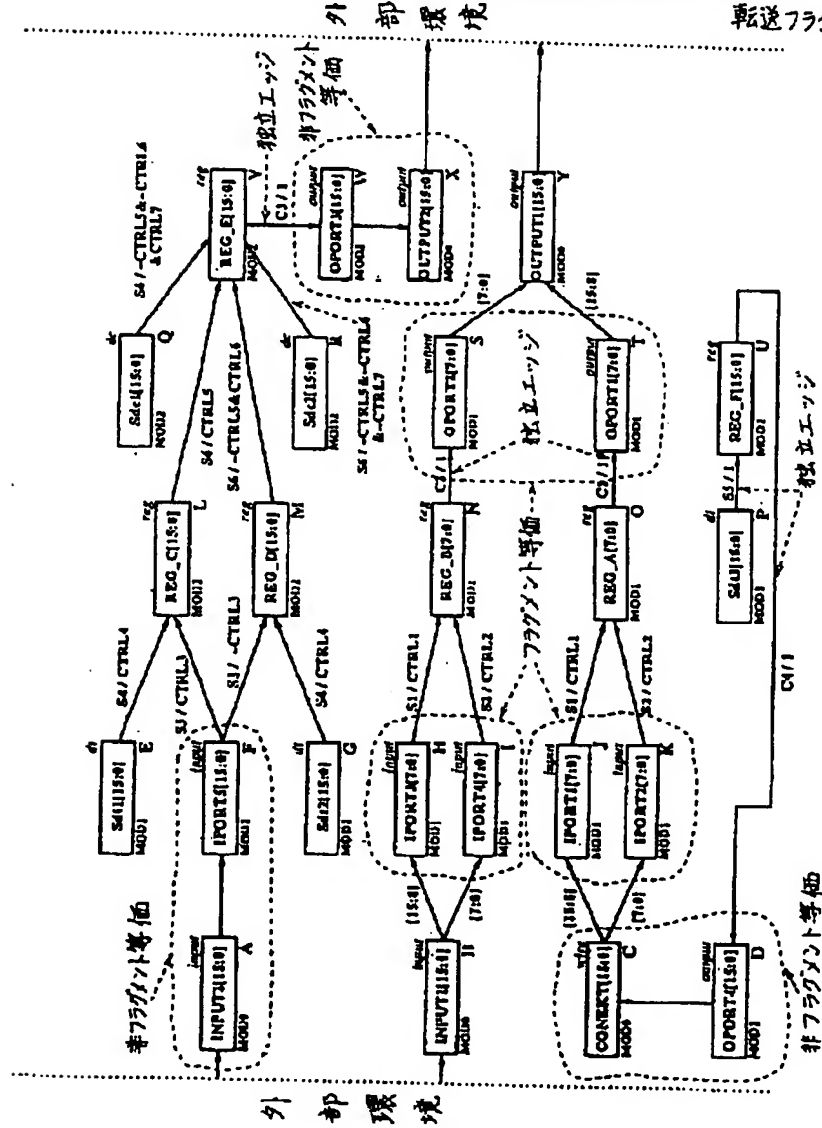
【図26】

Verilogで記述されたハードウェアは様の一例 (その3)



【図16】

図10～図15に示したハードウェア仕様から生成されたデータ
転送フラグ



【図21】

「資源競合」のプロパティの例

Resource Contention Properties

Node L: $AG(CTL4, CTL3)$
 Node M: $AG(CTL3, CTL4)$
 Node N: $AG(CTL3, CTL3)$
 Node O: $AG(CTL3, CTL3)$
 Node V: $AG(CTL3 \wedge \neg CTL5 \wedge \neg CTL7, CTL5)$

 For data transfer edges $E \in RL$
 For data transfer edges $FM \in QL$
 For data transfer edges $NH \in NL$
 For data transfer edges $JO \in KO$
 For data transfer edges $OV \in LV$

 For data transfer edges $OV \in MV$
 For data transfer edges $OV \in RV$
 For data transfer edges $LV \in MV$
 For data transfer edges $LV \in RV$
 For data transfer edges $MV \in RV$

 $AG(\neg CTL5 \wedge \neg CTL6 \wedge CTL7, \neg CTL5 \wedge \neg CTL6 \wedge \neg CTL7)$
 $AG(CTL5, \neg CTL3 \wedge CTL4)$
 $AG(CTL5, \neg CTL5 \wedge \neg CTL6 \wedge \neg CTL7)$
 $AG(\neg CTL5 \wedge CTL6, \neg CTL5 \wedge \neg CTL6 \wedge \neg CTL7)$

【図22】

「レジスタ漏れ」のプロパティの例

Register Leakage Properties

Node L: $AG_Read_L = \neg CTL5$

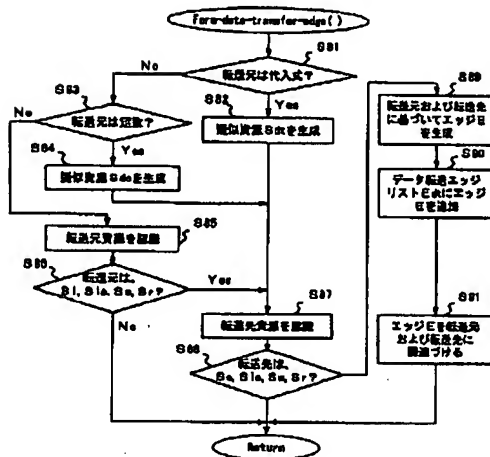
 For two consecutive writes from E to L
 For two consecutive writes from F to L
 For a write from F to L followed by a write from E to L
 For a write from E to L followed by a write from F to L

 Node M: $AG_Read_M = \neg(CTL5 \wedge CTL6)$

 For two consecutive writes from F to M
 For two consecutive writes from Q to M
 For a write from F to M followed by a write from Q to M
 For a write from Q to M followed by a write from F to M

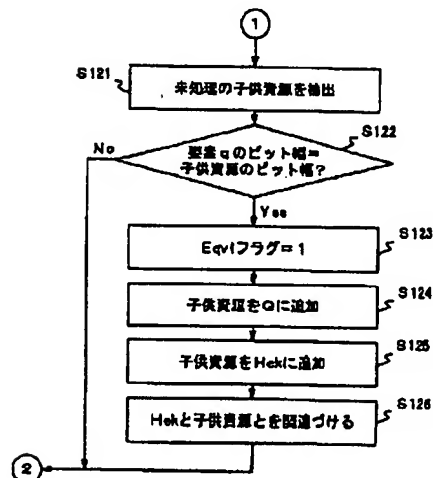
【図35】

データ転送エッジを生成する処理の詳細フローチャート



【図37】

「非フラグメント等価」を抽出する処理のフローチャート (その2)



【図25】

Verilogで記述されたハードウェア仕様の一例
(その2)

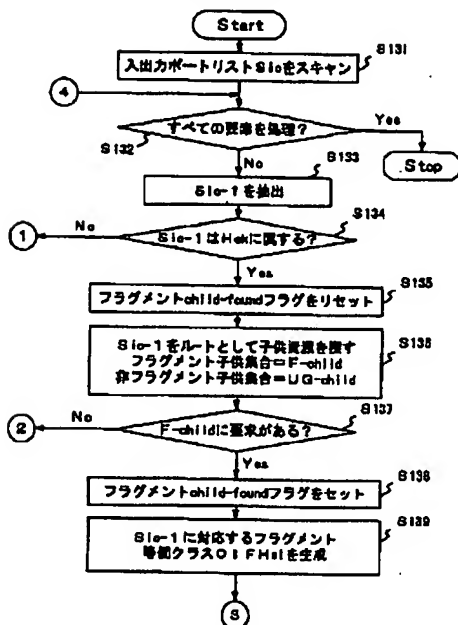
```

②
S1: always @(posedge GLOBAL_CLK)
begin
  if(CNT_RST_ACC <= WAIT0)
  begin
    case(RST_JUR(1))
    'b01: begin
      RST0(16:0) = EXT_DATA_OUT(31:16);
      RST0(7:0) = EXT_DATA_OUT(15:8);
      RST0(16:0) = EXT_DATA_OUT(15:8);
      RST0(7:0) = EXT_DATA_OUT(7:0);
    end
    'b11: begin
      RST0(16:0) = EXT_DATA_OUT(31:16);
      RST0(7:0) = EXT_DATA_OUT(23:16);
    end
    default:;
  end
endcase
end
end

```

【図38】

「フラグメント等価」を検出する処理のフローチャート(その1)



【図28】

図27に示すグラフにおけるデータ転送の条件を
示す図

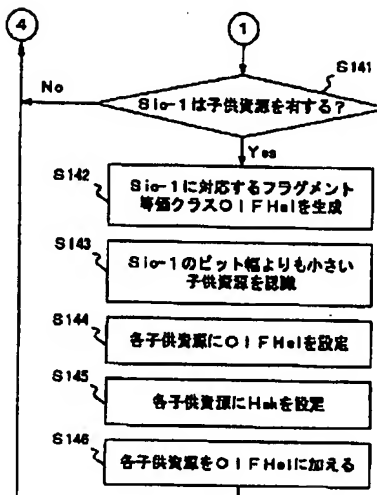
```

Guard/Enable Expressions in Defined ITDAG of streamout
G1 = CPT_RST_ACC & WAIT0 & RST_JUR(1) & RST_JUR(0)
G2 = CPT_RST_ACC & WAIT0 & RST_JUR(1) & RST_JUR(0)
G3 = CPT_RST_ACC & WAIT0 & RST_JUR(1) & RST_JUR(0)
G4 = CPT_RST_ACC & WAIT0 & RST_JUR(1) & RST_JUR(0)
G5 = CPT_RST_ACC & WAIT0 & RST_JUR(1) & RST_JUR(0)
G6 = CPT_RST_ACC & WAIT0 & RST_JUR(1) & RST_JUR(0)
G7 = COMPARE & WAIT0 & RST_JUR(1) & RST_JUR(0)
G8 = COMPARE & WAIT0 & RST_JUR(1) & RST_JUR(0)
G9 = COMPARE & WAIT0 & RST_JUR(1) & RST_JUR(0)
G10 = COMPARE & WAIT0 & RST_JUR(1) & RST_JUR(0)

```

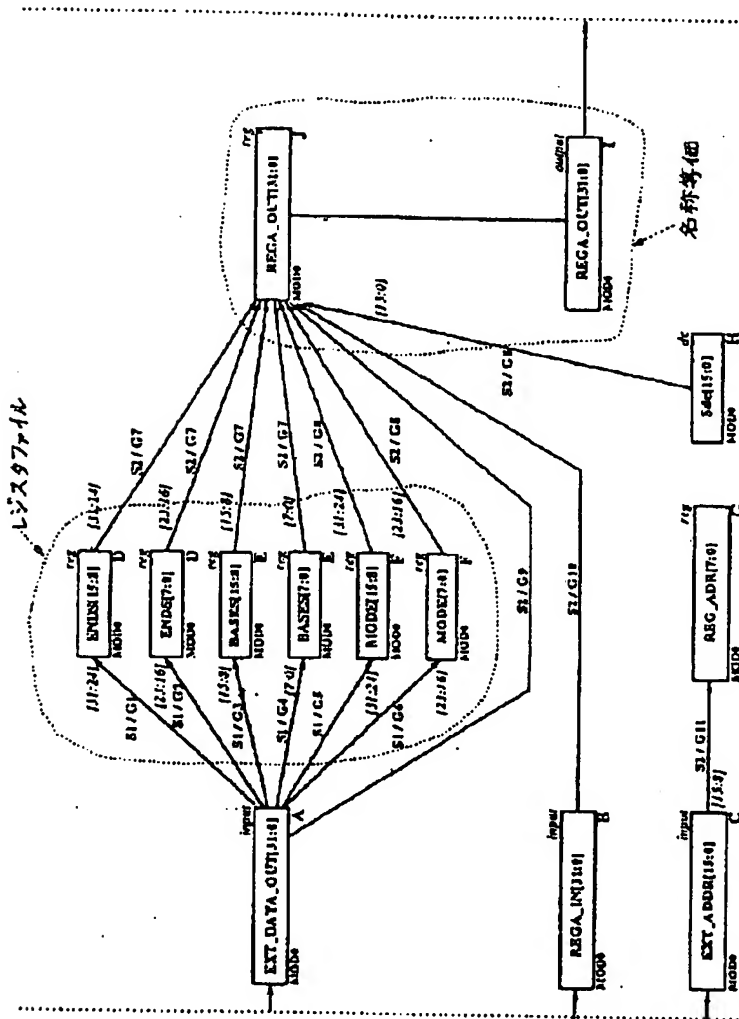
【図39】

「フラグメント等価」を検出する処理の
フローチャート(その2)



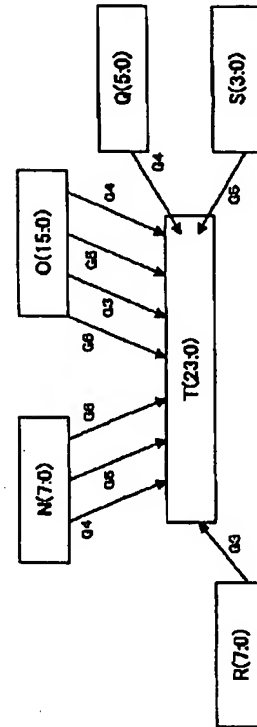
【図27】

図24～図26に示したハードウェア仕様から
生成されたデータ転送グラフ



【図49】

データ転送グラフの例



【図29】

レジスタファイルを検出する処理を説明する図

Register File Identification Stage

```

Support(C1) = ( CMT_ACC, WRITE, REG_ADR(4), REG_ADR(3), REG_ADR(2), REG_ADR(1), REG_ADR(0) )
Support(C2) = Support(C1) + Support(C3) = Support(C4) = Support(C5) = Support(C6)
Sink_Reg_File = ( REGIS(15:0), VALUE(15:0), MODE(15:0) )
Support(C7) = ( COMPARE, WRITE, CMT_ACC, REG_ADR(7), REG_ADR(6), REG_ADR(5), REG_ADR(4) )
Support(C8) = Support(C7)
Source_Reg_File = ( REGIS(15:0), VALUE(15:0), MODE(15:0) )
As, Source_Reg_File = Sink_Reg_File.
Reg_File = ( REGIS(15:0), VALUE(15:0), MODE(15:0) )

```

【図31】

最適化されたグラフにおいて
使用される条件を示す図

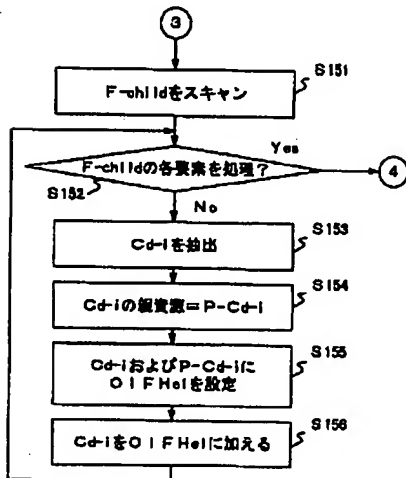
New Guard/Enable Expressions in Optimized DTRAG of example2

```

O1' = O1 | O3 | O4 | O5 | O6
O1' = CMT_ACC & WRITE
O3' = O7 | O8
O3' = COMPARE & WRITE & CMT_ACC & REG_ADR(7) & REG_ADR(6) & REG_ADR(5)

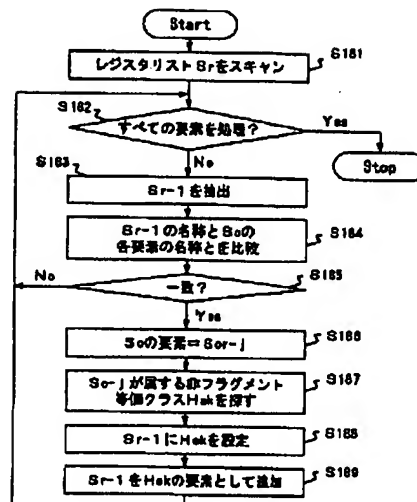
```

【図40】

「フラグメント等価」を検出する処理の
フローチャート (その3)

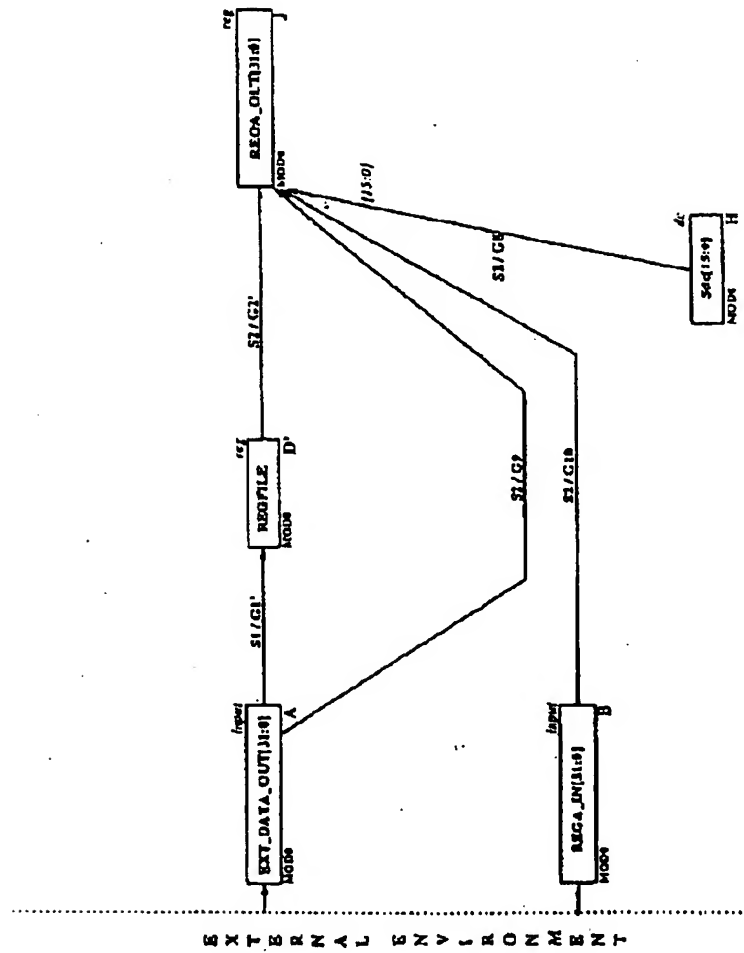
【図42】

「名称等価」を検出する処理のフローチャート



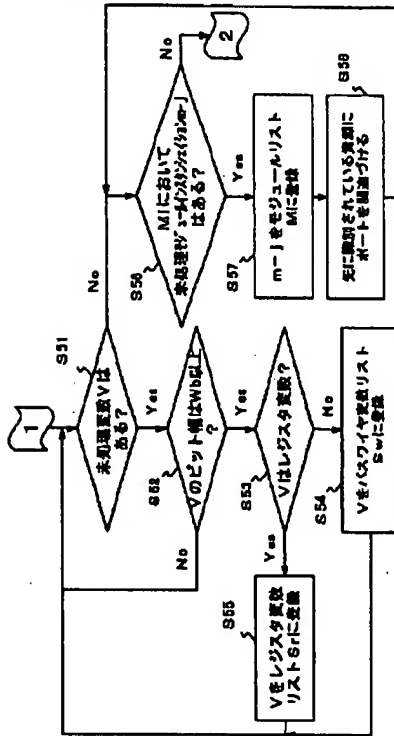
【図30】

最適化されたデータ転送グラフの例



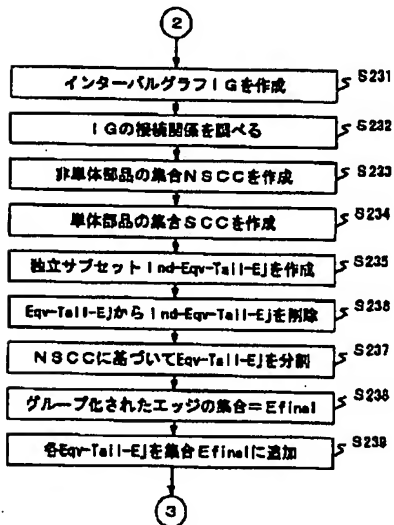
【図 3 3】

データ転送グラフを生成する方法のフローチャート(その2)



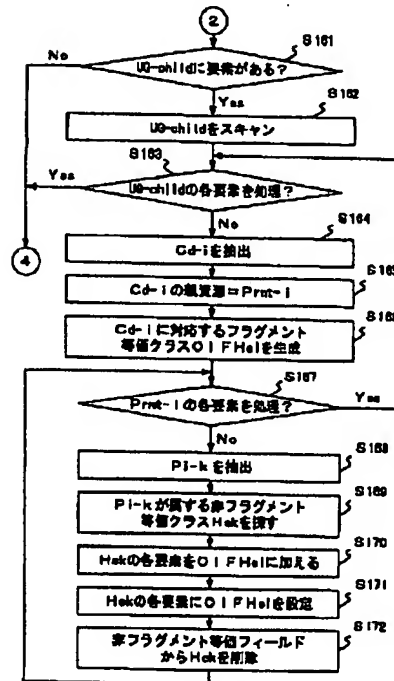
【図45】

「等価エッジ」を検出する処理の
フローチャート（その2）



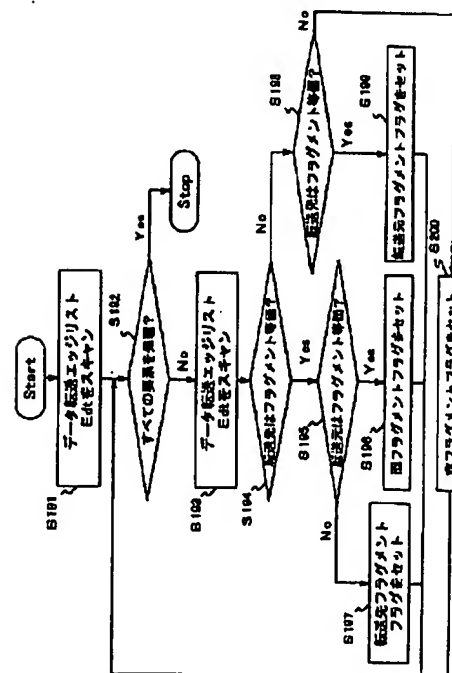
【図 4 1】

「フラグメント野値」を検出する処理のフローチャート（その4）



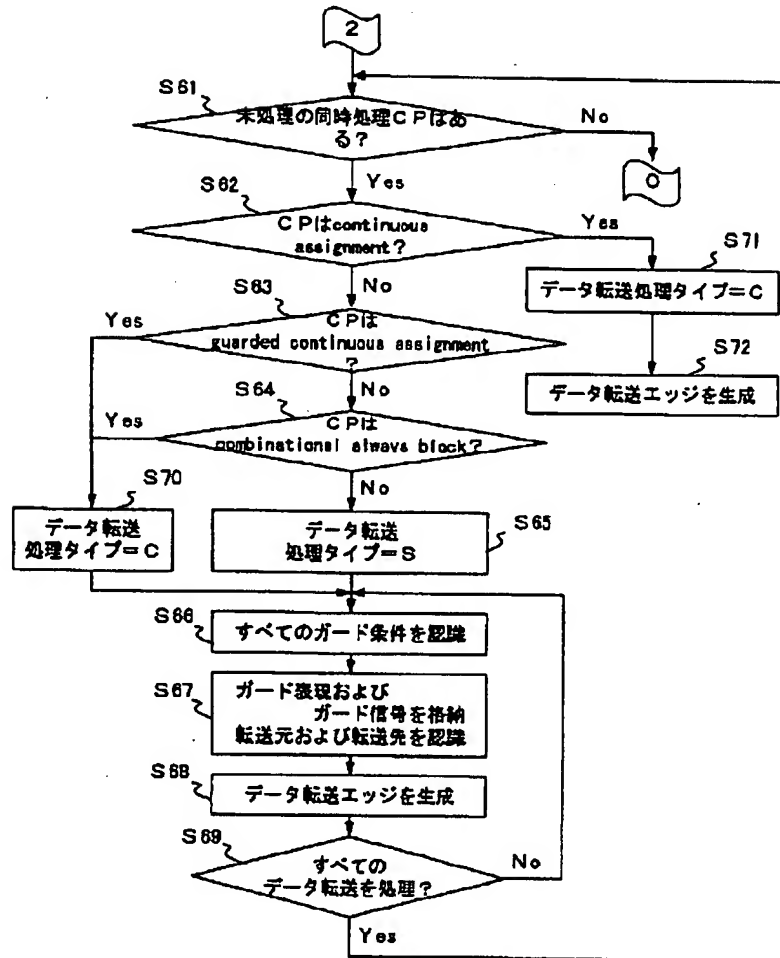
【図43】

「分岐エッジ」を検出する処理のフローチャート



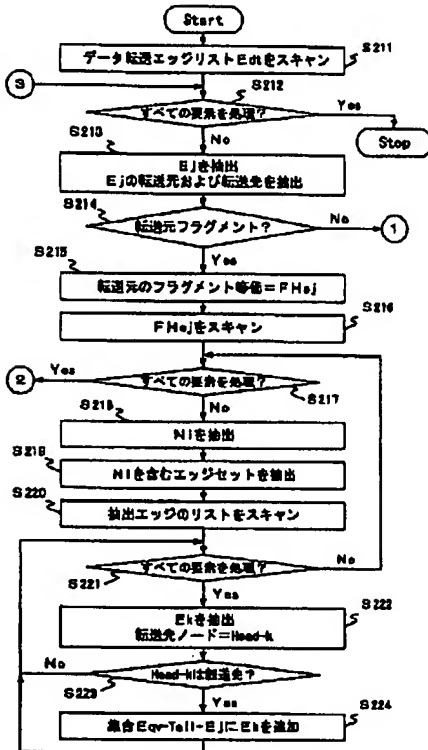
【図34】

データ転送グラフを生成する方法のフローチャート（その3）



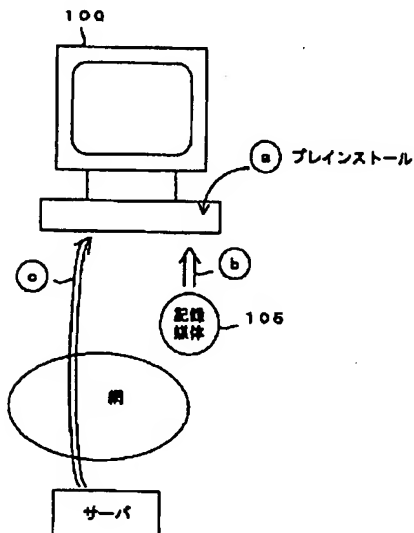
【図44】

「等価エッジ」を検出する処理のフローチャート（その1）



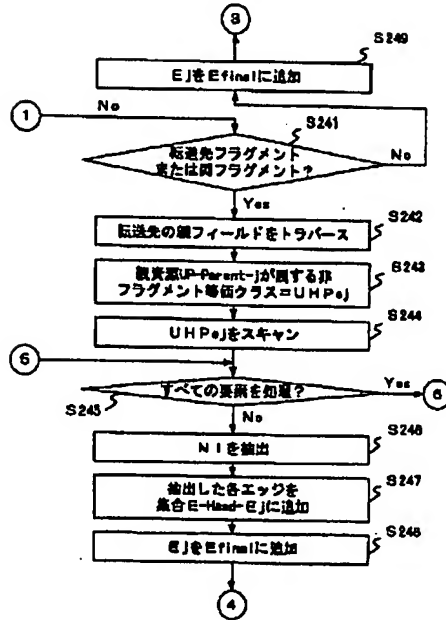
【図62】

本発明に係わるソフトウェアプログラムなどの提供方法を説明する図



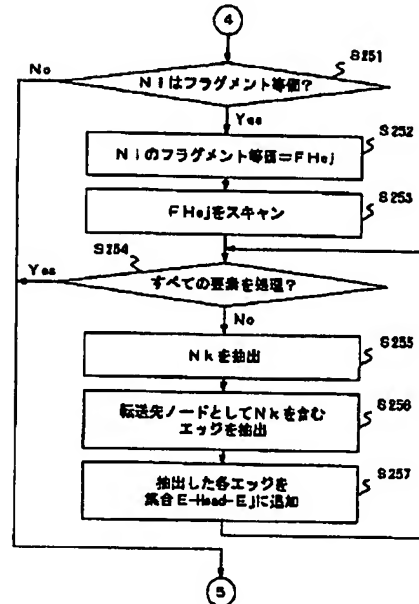
【図46】

「等価エッジ」を検出する処理のフローチャート（その3）



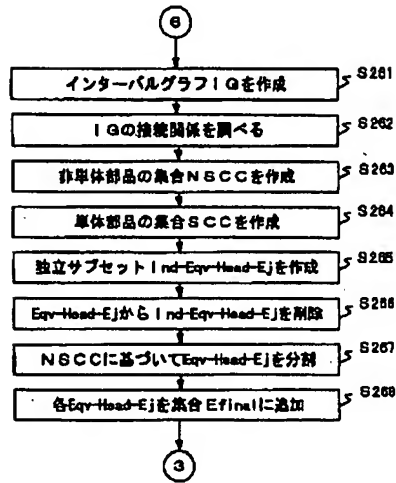
【図47】

「等価エッジ」を検出する処理のフローチャート（その4）



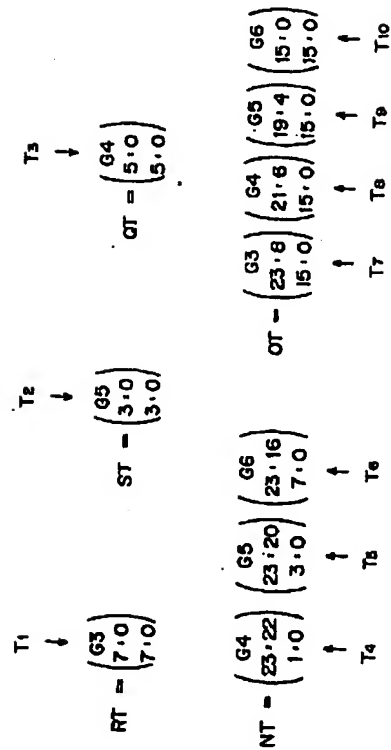
【図48】

「等価エッジ」を検出する処理の
フローチャート（その5）



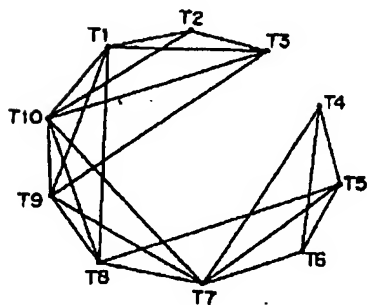
【図50】

データ伝達の定義



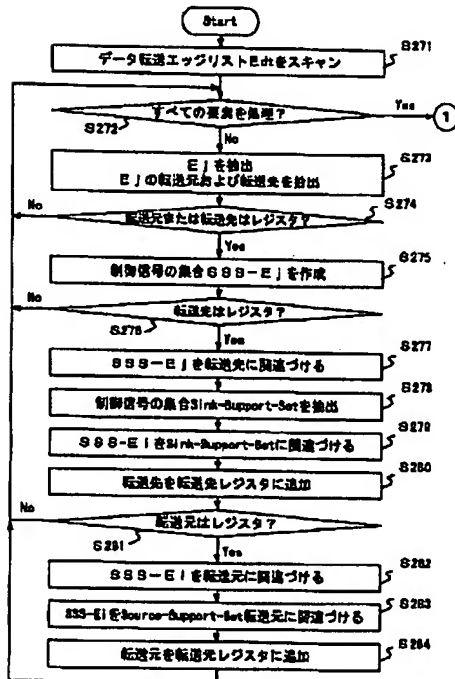
【図51】

インターバルグラフの例



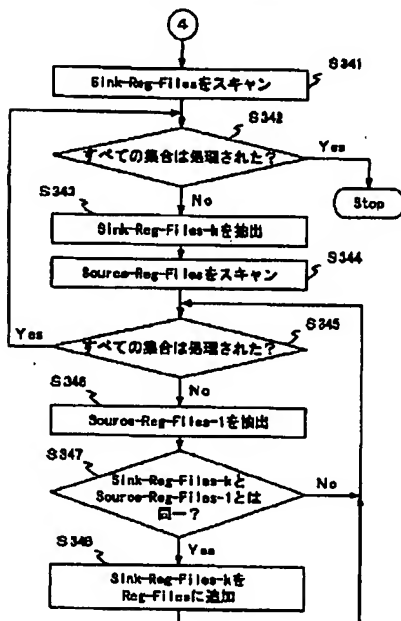
【図52】

「レジスタファイル」を検出する処理のフローチャート（その1）



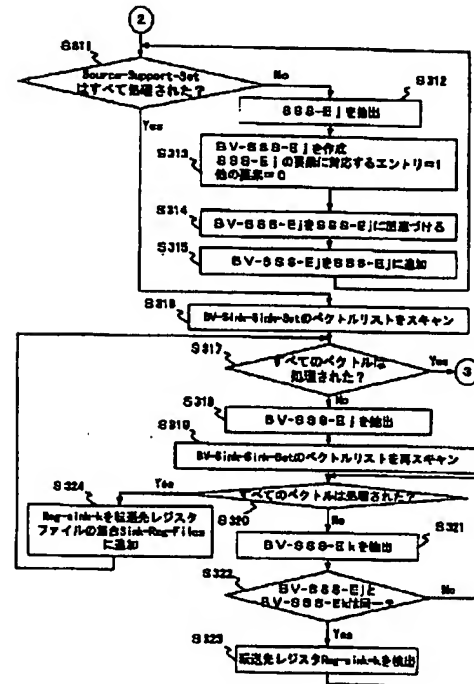
【図56】

「レジスタファイル」を検出する処理のフローチャート（その5）



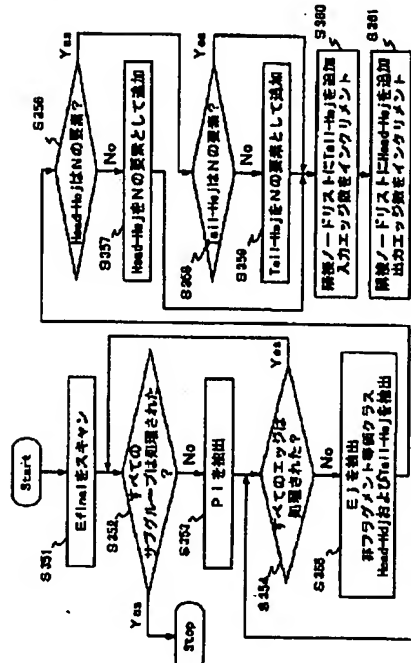
【図54】

「レジスタファイル」を検出する処理のフローチャート（その3）



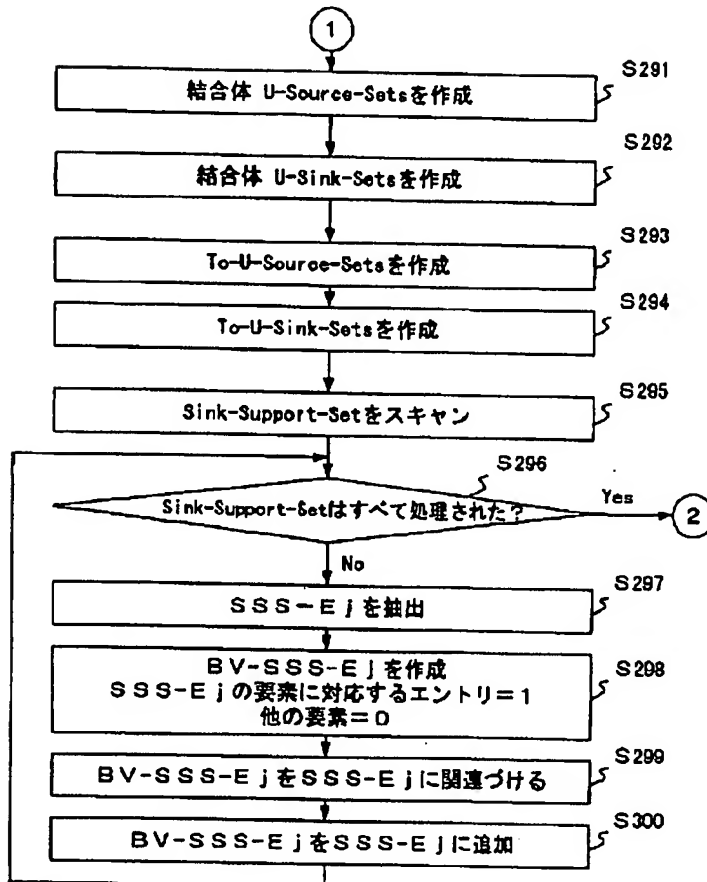
【図57】

隠性ノードリストを作成する処理のフローチャート



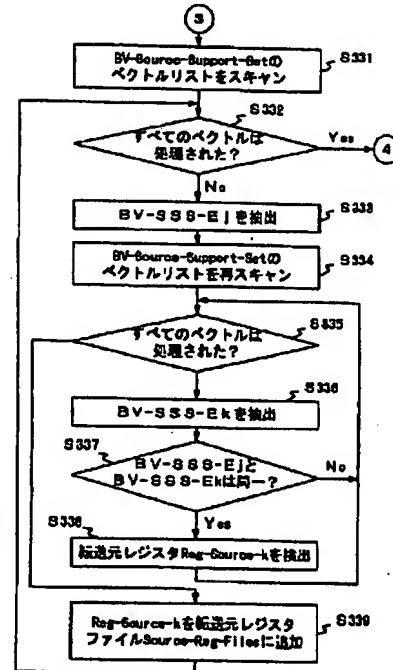
【図53】

「レジスタファイル」を検出する処理のフローチャート（その2）



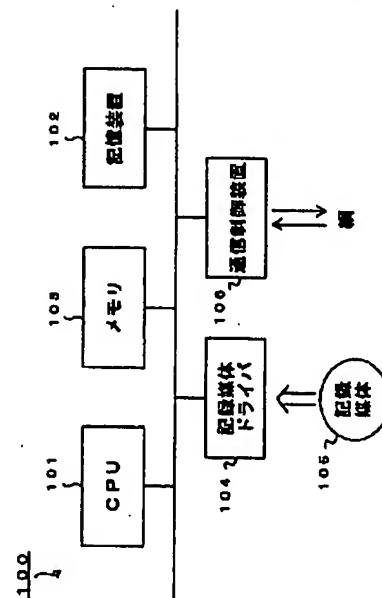
【図55】

「レジスタファイル」を検出する処理のフローチャート（その4）



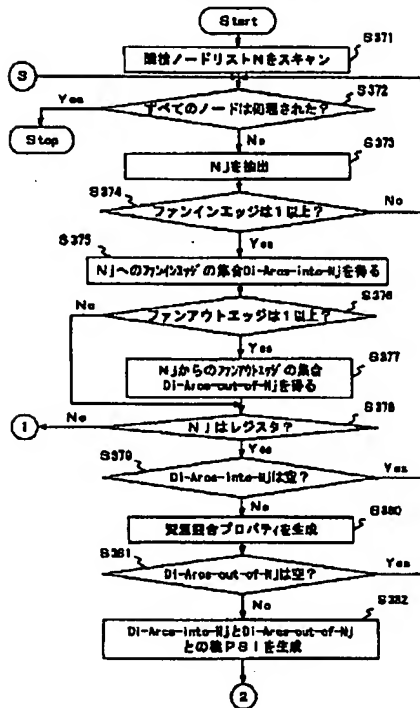
【図61】

本発明の機能を記述したプログラムを実行するコンピュータのブロック図



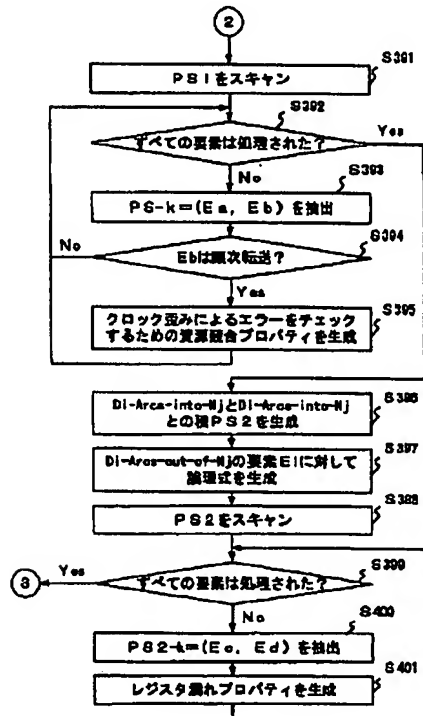
【図58】

プロパティスクリプトを生成する処理のフローチャート（その1）

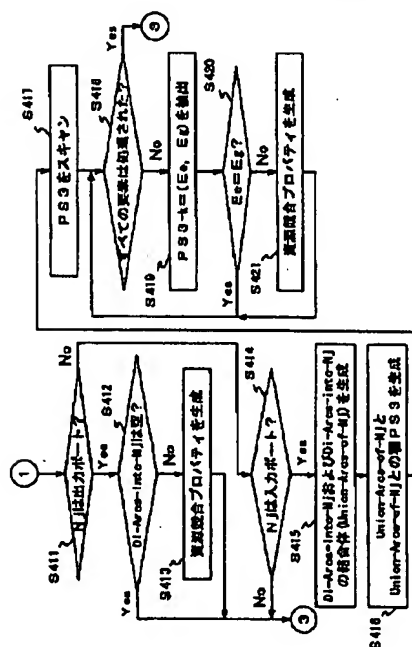


【図59】

プロパティスクリプトを生成する処理のフローチャート（その2）



プロパティスクリプトを生成する処理の
フローチャート(その3)



U

Fターム(参考) 5B046 AA08 DA04 DA06 JA01
5F038 CA03 CA17 CD05 DF11 EZ10
EZ20
5F064 BB09 BB18 DD03 DD04 DD25
HH06 HH09 HH10 HH14